FLAVIUS PORTELLA RIBAS MARTINS

CALIBRAÇÃO DE PADRÕES METROLÓGICOS UTILIZANDO VISÃO COMPUTACIONAL

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia

SÃO PAULO

1999

FLAVIUS PORTELLA RIBAS MARTINS

CALIBRAÇÃO DE PADRÕES METROLÓGICOS UTILIZANDO VISÃO COMPUTACIONAL

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia

Área de Concentração:
ENGENHARIA MECÂNICA

Orientador:
FRANCISCO EMÍLIO BACCARO NIGRO

SÃO PAULO

1999

A minha adorada filha Laura

Agradecimentos

Este trabalho não poderia ter sido realizado não fosse a enorme cooperação que eu obtive, de pessoas e instituições, para poder dar-lhe conteúdo e forma. Assim sendo, gostaria de apresentar os meus sinceros agradecimentos

ao Dr. Francisco Nigro, meu orientador, pelo entusiástico apôio que me prestou e por suas preciosas observações, sugestões e análises que não apenas enriqueceram, mas, principalmente, deram fundamento a etapas essenciais deste trabalho;

aos Engºs Walter Link e Marisa Ferraz Figueira, pelos esclarecimentos prestados no âmbito específico da metrologia e pela disponibilização de todos os recursos do Laboratório de Metrologia do IPT necessários à execução deste trabalho;

aos técnicos do Laboratório de Metrologia do IPT, Carlos Brokhof e Douglas Yamanaka, pelo esforço devotado à realização dos diversos experimentos metrológicos solicitados por este autor;

ao pessoal do Laboratório de Óptica do IPT, pelas inúmeras dúvidas dirimidas sobre aspectos fundamentais da instrumentação óptica;

ao Engº Saburo Ikeda que, então chefe da Divisão de Mecânica e Eletricidade do IPT e consciente do impacto positivo que a automação poderia ter sobre o desempenho do Laboratório de Metrologia, empenhou seu esforço pessoal para que este trabalho pudesse ser realizado;

a todos os colegas da Divisão de Mecânica e Eletricidade do IPT, pelo constante apôio e estímulo;

ao IPT como um todo, pelas facilidades oferecidas durante o desenvolvimento das diversas etapas do programa de doutorado da EPUSP.

RESUMO

Em processos metrológicos, tradicionalmente se ignora o efeito das escalas nominal e ordinal sobre a qualidade final das medidas. Tais escalas, por estarem ligadas respectivamente a reconhecimento e classificação de padrões, atividades desenvolvidas inconscientemente pelo operador humano, não despertam qualquer atenção no processo de medição, onde todos os esforços se concentram sobre a escala racional, descritora das medidas numéricas. Contudo, durante a automação de ensaios de calibração de microescalas de 1mm e microtexturas de peneiras de medição, a modelagem do conhecimento visual humano do operador e, portanto, dos processos de reconhecimento e classificação de padrões, mostrou-se crucial. Após se construir sete instrumentos metrológicos baseados em diferentes paradigmas de visão computacional, realizaram-se experimentos que, além de demonstrarem de forma inequívoca a influência das escalas nominal e ordinal sobre a qualidade das medidas finais, permitiram identificar, para os dois ensaios considerados, os modelos de medição e respectivos parâmetros que produziam medidas com maior repetibilidade e exatidão. Os dois modelos identificados, foram, então aplicados, à medição de microescalas e microtexturas de peneiras, obtendo -se, relativamente aos ensaios convencionais, diminuição da incerteza de até 5 vezes (0,2μm para 0,04μm) no primeiro caso e de até 20 vezes (2μm para 0,2μm) no segundo. Já o ganho em eficiência foi ainda mais significativo: os instrumentos desenvolvidos para calibrar microescalas e microtexturas mostraram ser capazes de processar a mesma quantidade de informação, respectivamente, em intervalos de tempo 30 e 100 vezes inferiores ao requerido pelos processos metrológicos tradicionais.

ABSTRACT

Traditionally, the analysis of metrological processes completely ignore the effect of the so called nominal and ordinal scales on the quality of the measurements. While such scales are concerned, respectively, with pattern recognition and classification, the focus of attention in typical measurements is the numerical results represented through the rational scale. However, during a project developed to implement automation on two metrological procedures — calibration of 1 mm linear precision rules and calibration of wire-cloth sieves for testing purpose — the act of measuring on those scales had to be considered, since it was necessary to represent and model the non-structured human vision knowledge used unconsciously by the metrologist to understand images and operate instruments. After constructing a group of seven software-based metrological instruments, using different computer vision paradigms, a series of experiments demonstrated clearly the influence of nominal and ordinal scales on the repeatability and accuracy of the measurements. Furthermore, through such experiments it was identified the best computer vision tools and their correspondent parameters to be assumed henceforth as the software-based instruments to measuring rules and wire-cloth sieves. Finally, after a new series of measurements, using these specific instruments, their performance were characterized comparatively with the traditional metrological procedures, and the following results were ascertained: to calibrate rules, the necessary time and the uncertainty in measurements decreased respectively 5 times and 20 times; concerning wire-cloth sieves calibration, the results were even more significant: 20 times less uncertainty and up to 100 times more efficiency to process the same amount of information.

Errata

Na página i (Resumo), onde se lê *2μm para 0,2μm* leia-se *2μm para 0,1μm.*

Nas páginas 1, 121, 139 e 162, onde se lê *a nível* leia-se *ao nível.*

Nas páginas 3,4,5,10, 13, onde se lê *teoria da medida* leia-se *teoria da medição.*

Na página 6, onde se lê *pastilha de vidro* leia-se *pastilha de aço inox.*

Na página 6, onde se lê *disparando o intereferômetro* leia-se *ativando-se o interferômetro.*

Na página 68, onde se lê *em sua forma mais utilizada...*, leia-se, *em sua forma mais utilizada no âmbito do problema de identificação e localização de retas...* .

Na página 115, acrescentar após a sentença *"...e imediatamente após são apresentados os demais experimentos de avaliação de desempenho dos instrumentos implementados"*, a seguinte oração: *"onde a técnica Projeto de Experimentos não foi utilizada, de vez que existia suficiente conhecimento de determinadas dependências entre as diversas variáveis dos referidos instrumentos"* .

Na página 119, o sexto algarismo das medidas apresentadas nas colunas 4 a 7 não são significativos.

Na página 120, o sexto algarismo das medidas apresentadas nas colunas 5 a 7 não são significativos.

Na página 137, onde se lê *±0,00004μm*, leia-se *±0,00004mm.*

Na página 151, onde se lê *da ordem de 0,00005μm* leia-se *da ordem de 0,00005mm.*

Na página 160, onde se lê *±0,00004μm*, leia-se *±0,00004mm.*

SUMÁRIO

Lista de Figuras

Lista de Tabelas

1 - INTRODUÇÃO

Duas grandes vertentes da metrologia dimensional têm suas origens situadas no fim do século XVIII. Uma, a metrologia de padrões primários, é o produto das mentes racionalistas da Revolução Francesa. A outra, a metrologia de padrões secundários, é fruto da engenhosidade de um representante da Revolução Industrial — Eli Whitney.

A constatação de que a existência de múltiplas unidades de medida definidas sem qualquer critério científico trazia dificuldades consideráveis ao comércio, levou o governo revolucionário francês a nomear uma comissão de engenheiros para estudar o problema. Desse trabalho originou-se o sistema métrico, cuja unidade primária, o metro, era definido com base em um padrão geográfico — "a décima milionésima parte do arco de 45° do meridiano passando através do Observatório de Paris." À mesma época, Eli Whitney recebe do governo americano a incumbência de fabricar 15000 mosquetes cujos componentes pudessem ser intercambiados. O problema era complexo pois, apesar de já então se utilizarem máquinas operatrizes, os sistemas de produção contemporâneos baseavam-se em métodos eminentemente artesanais. Os processos produtivos de Whitney, utilizando padrões de referência especialmente desenvolvidos para a fabricação de componentes seriados, constituiram a pedra fundamental da metrologia de padrões secundários e, repetindo as palavras de MOORE (1970), "o principal fator que levou os Estados Unidos a rapidamente adquirirem proeminência industrial sobre as demais nações".

Dois séculos de contínua evolução, tanto a nível de processos quanto a nível da linguagem, transformaram a prática da medição em uma ciência — a ciência da metrologia, cujas principais missões podem ser assim resumidas:

1. Definir e manter um sistema coerente de unidades — os chamados padrões primários;
2. Definir e calibrar padrões de referência para a indústria — os chamados padrões secundários;
3. Elaborar normas e processos de medição robustos, precisos e confiáveis;
4. Efetuar medições em componentes e sistemas industriais.

A organização e manutenção dos padrões primários é tema exclusivo das Instituições Internacionais de Metrologia, que periodicamente discutem e revisam as definições das unidades primárias, de modo que elas expressem a máxima resolução proporcionada pela instrumentação metrológica disponível no estado da arte. No outro extremo da cadeia metrológica encontram-se os técnicos da indústria, desenvolvendo atividades variadas de alinhamento e verificação de dimensões e tolerâncias de ítens acabados, utilizando padrões secundários e processos de medição desenvolvidos por metrologistas. Entre os dois extremos situam-se as redes de laboratórios de padrões secundários credenciados pelos Institutos Nacionais de Metrologia, desenvolvendo as missões citadas nos ítens 2 e 3 acima, objeto deste trabalho. Essas redes são elementos-chave para o aumento da competitividade industrial, visto que, através da precisa calibração de padrões de referência, estabelecem a base para implantação de processos de manufatura capazes de atender aos requisitos de qualidade de mercados cada vez mais exigentes.

A metrologia dimensional, por se tratar de uma ciência voltada para a tecnologia, sempre se utilizou dos recursos mais avançados, ora requeridos para aprimorar o sistema de medidas, ora para atender às novas necessidades do setor produtivo. É importante notar que essas demandas, vindas de ambos os extremos da cadeia metrológica, provocam impacto significativo nos laboratórios de padrões secundários, os quais se vêm compelidos a desenvolver novos sistemas de instrumentação e processos

compatíveis de medição e tratamento de dados para atender àquelas demandas. Abaixo apresentam-se dois exemplos importantes que ilustram a interação entre esses três elos.

Em 1905, oriunda dos laboratórios de padrões primários, a medição do metro-padrão em função do comprimento de onda da linha vermelha do espectro de cádmio transformou radicalmente a instrumentação metrológica. Desde então, variados métodos baseados em fenômenos de interferência óptica têm sido desenvolvidos para aplicações típicas de laboratórios de padrões secundários, tais como: medição de rugosidade e planeza, através de métodos de projeção de franjas de Moiré e calibração de blocos-padrão e microescalas, através de métodos de interferometria clássica (THOMAS,1974).

Oriunda da indústria, a máquina de medição de coordenadas provocou, nos anos 80, uma tal revolução nos princípios e conceitos que norteavam a metrologia, que se chegou a falar em uma "crise da metrologia". O que ocorria é que, enquanto se desenvolvia um instrumento de medição capaz de gerar uma enorme massa de dados descritores de forma, as normas e processos de medição continuavam atreladas a conceitos e princípios antigos, remanescentes do período em que comprimentos eram medidos com paquímetro ou micrômetro (SURESH; VOELCKER, 1994). Em outras palavras, a tecnologia de medição estava muito além do que a limitada semântica da linguagem metrológica vigente era capaz de expressar, favorecendo múltiplas interpretações para os dados, logo, inconsistência nas medidas. Nesse cenário de crise, os laboratórios de padrões secundários foram capazes não apenas de diagnosticar a sua causa primordial (descompasso entre linguagem e tecnologia) como também passaram a se dedicar à proposição de soluções que permitissem compatibilizar os avanços tecnológicos com os fundamentos dos processos de medição. Frutos importantes desse trabalho que, aliás, continua em andamento, são o surgimento da metrologia computacional e a revalorização da teoria da medida.

HOFMANN (1994) sintetiza a impressão atual e consensual entre metrologistas quando afirma que o desempenho e flexibilidade dos novos instrumentos de medição dependem essencialmente do software. Apesar de constituírem o cerne de um grande número de instrumentos metrológicos modernos, na maioria dos casos os algoritmos de aquisição e tratamento de dados operam como "caixa preta", dificultando a avaliação da qualidade das medições efetuadas. Em tais circunstâncias, fica evidente a necessidade de uma meta-metrologia — a "metrologia do software de metrologia" ou metrologia computacional (HOPP, 1993). Da mesma forma que na metrologia clássica, as atividades dessa nova disciplina abrangem o desenvolvimento de instrumentos e processos de medição e tomada de ações de carácter normativo. No primeiro caso, através do projeto de softwares para a identificação de parâmetros de desempenho de softwares de medição do tipo "caixa preta"; no segundo caso, fornecendo subsídios para a construção de softwares de medição cujo desempenho possa ser facilmente avaliado por meio de testes padronizados.

A teoria da medida, originalmente voltada ao tratamento de problemas de informação e controle, vem desempenhando um papel relevante na descrição dos aspectos semânticos dos processos metrológicos, contribuindo para que eles acompanhem *pari passu* os avanços tecnológicos do instrumental de medição. À luz da teoria da medida, evidencia-se a complexidade de conceitos que antes eram muitas vezes tratados como simples axiomas. A título de ilustração, considere-se o conceito fundamental *medição* e sua abordagem, inicialmente, pelos autores clássicos de metrologia. MILLER (1962), ao definir metrologia como "a ciência de medir com precisão," logo esclarece que o termo *precisão* se refere a "uma tentativa de determinar o valor verdadeiro de uma grandeza", mas não atribui qualquer significado específico ao termo *medir*. BECKWITH *et al* (1981), por seu turno, consideram *medição* como

"um simples processo de comparação entre a grandeza a ser medida e um padrão", opinião também compartilhada por BUSCH (1985), que afirma que "medição é sempre comparação — comparação entre uma distância conhecida e outra desconhecida". Sob a óptica dos adeptos da teoria da medida, o conceito *medição* adquire novos contornos e gera considerável polêmica. FILKENSTEIN (1994) define *medição* como um processo informativo e empírico que associa números ou outros símbolos a atributos de componentes ou eventos de um sistema. Para FIOK *et al* (1991), *medição* é "um experimento de identificação de parâmetros de um modelo matemático do objeto ou fenômeno a ser medido", opinião contestada por ABDULLAH *et al* (1994) segundo os quais aquela definição apenas sintetiza o propósito da teoria de identificação de sistemas; *medição* identifica parâmetros sim, porém utilizando um modelo *a priori* contendo conhecimento teórico-experimental sobre o sistema a ser medido. HOFMANN (1991; 1994), de certa forma encerra a polêmica, quando afirma categoricamente que a exata definição de *medição* é um dos problemas ainda não resolvidos no âmbito da metrologia.

Os princípios da metrologia computacional e os novos conceitos de medição derivados da teoria da medida encontram campo fértil de aplicação em projetos de instrumentos metrológicos baseados em visão computacional. Conforme se apresenta em (BALLARD; BROWN, 1982), o objetivo da visão computacional é construir modelos cognitivos que representem cenas do mundo real. Combinando conhecimento *a priori* da cena observada com técnicas de processamento de imagens, reconhecimento de padrões e ajuste de parâmetros, os sistemas de visão computacional devem gerar informações invariantes a partir de imagens que apresentem variações irrelevantes do ponto de vista cognitivo, tais como efeitos de iluminação, translações, rotações e mudanças de escala. Essa tecnologia já vem sendo utilizada em instrumentos de

inspeção e medição industrial há bem mais de uma década, todavia, a sua incorporação à instrumentação dos laboratórios de metrologia de padrões secundários é fato recente, pois também é muito recente a pesquisa acerca das características de precisão dos sistemas de aquisição de imagens e dos algoritmos de visão computacional.

Dentre a variada gama de ensaios realizados em laboratórios de padrões secundários, existe uma categoria deles para os quais instrumentos de medição baseados em visão computacional seriam de grande valor: trata-se dos ensaios em que o operador realiza a medição observando e monitorando uma determinada imagem através de um instrumento óptico. Atuando sobre os dispositivos de deslocamento mecânico deste instrumento e demais comandos porventura existentes, o operador efetua a coleta de dados no instante em que a imagem apresenta propriedades que ele julga adequadas para a medição. Calibração de microescalas e calibração de microtexturas de peneiras de medição são dois exemplos clássicos de ensaios apresentando essas características.

Microescalas são padrões secundários de referência utilizados para a calibração de microscópios industriais. Consistem de uma pequena pastilha de vidro sobre a qual é impressa uma régua bastante precisa com comprimento total de 1 ou 5 mm e divisões de 10μm. A instrumentação tradicional utilizada para a calibração desses padrões é composta por microscópio de medição e interferômetro com refletor montado sobre a platina do microscópio (figura 1.1). Nesse ensaio, o operador observa a imagem amplificada da microescala e, após orientar os seus traços na direção do cursor, utiliza o micrômetro para alinhar o cursor e um traço da microescala, disparando o interferômetro a cada novo alinhamento e registrando o deslocamento da platina e, portanto, a posição do traço no sistema fixo de coordenadas.

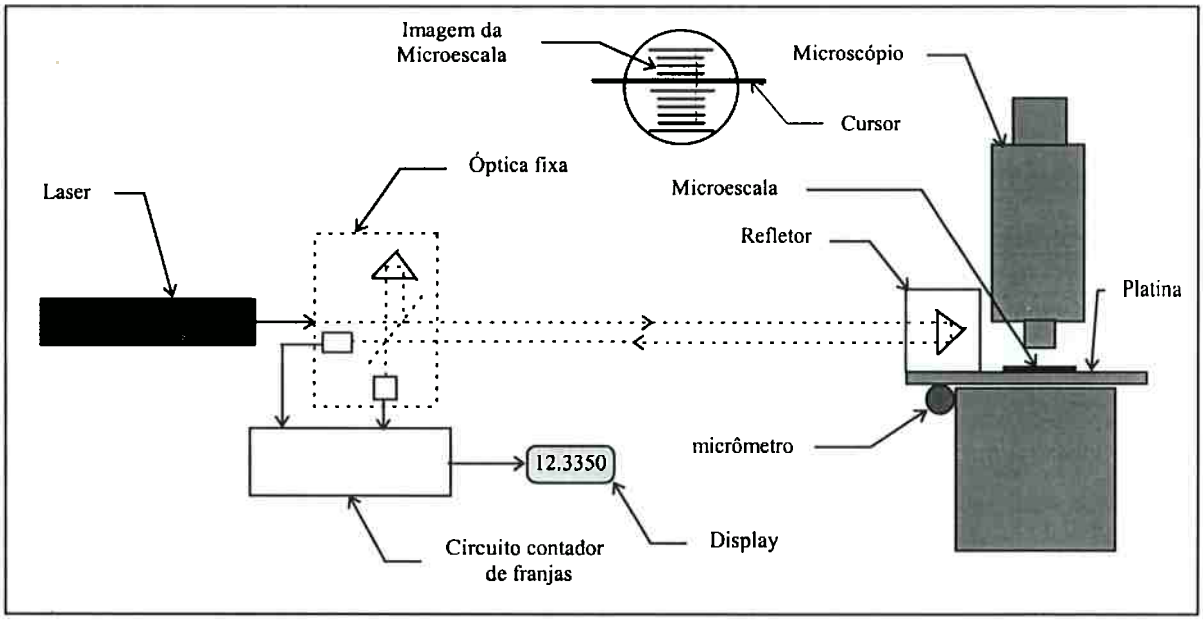

Figura 1.1: Arranjo experimental clássico para calibração de microescalas.

Peneiras de medição são padrões de referência utilizados pela indústria em ensaios de separação de materiais granulados, segundo o diâmetro médio dos grãos. As dimensões da abertura da trama dessas peneiras, descritas em normas internacionais, variam entre 20μm e 125mm. No caso da norma brasileira de peneiras para ensaio (ABNT-5734, 1991), a gama de dimensões é um pouco menor, variando entre 37μm e 108mm. Peneiras com aberturas não superiores a 15 mm são normalmente calibradas utilizando-se um projetor de perfis. Com esse instrumento, típico de laboratórios metrológicos, o operador mede as características da trama atuando sobre os três graus de liberdade (X, Y, θ) da mesa posicionadora, de modo a alinhar uma borda da sombra projetada com o cursor da tela. Completado o alinhamento, e sempre orientando-se pela sombra projetada da peneira, o operador vai obtendo, uma a uma, as medidas das aberturas e diâmetros dos fios, a partir do deslocamento linear (X-X ou Y-Y) da mesa (figura 1.2).

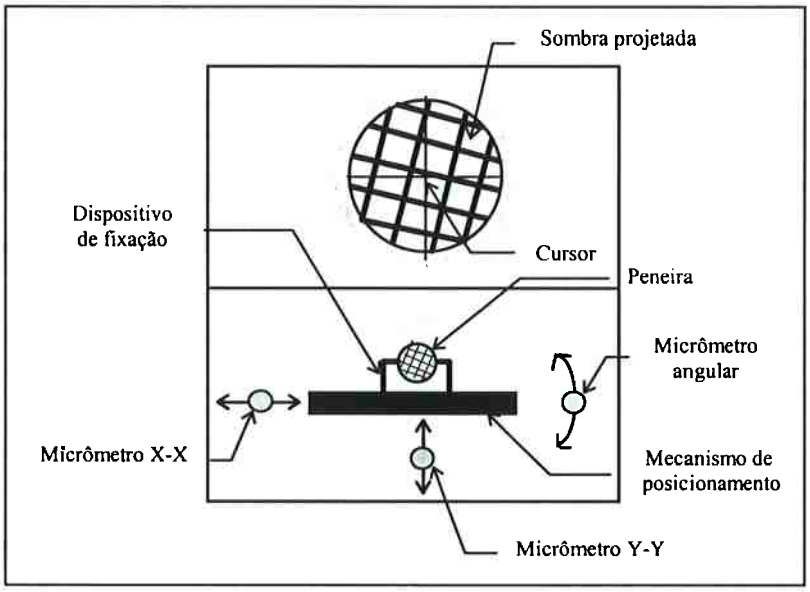

Figura 1.2: Calibração de peneira de medição usando projetor de perfis.

Rigorosamente, o que há de especial nesses dois ensaios é o fato de que a observação do padrão requer ampliação óptica e a imagem ampliada preserva as características métricas relevantes do padrão original. Portanto, a cena de interesse é adequadamente representada por uma imagem, o instrumental necessário para produzi -la já é parte integrante do ensaio e, ademais, existe um modelo *a priori* do objeto que se pretende medir, condições essenciais que possibilitam desenvolver algoritmos de visão computacional capazes de substituir as funções de observação, monitoração, decisão e comando executadas pelo técnico metrologista.

É importante aqui frisar que o papel desempenhado pelo operador humano nos processos de medição é tão essencial, que até mesmo experimentos metrológicos elementares seriam inconcebíveis sem a sua presença. Considere-se, por exemplo, o caso trivial de se medir com uma régua o lado maior de uma chapa retangular. Como o operador possui conhecimento prévio sobre o instrumento de medição e sobre a forma do objeto a ser medido, a sua atenção irá se concentrar apenas nos aspectos essenciais da cena, ou seja, régua e chapa. Fixando a chapa e movimentando a régua de forma a

alinhá-la com a direção do lado maior, ele monitora a cena até que uma das extremidades da chapa coincida com o traço zero da escala; neste momento, toma a decisão de efetuar a medida, "comandando" a régua para que ela forneça o valor do comprimento, ou seja, interpolando visualmente a posição da outra extremidade da chapa relativamente aos traços da escala.

Por causa dessa extrema dependência do operador, os instrumentos metrológicos são projetados para utilizarem, da melhor forma possível, certas capacidades sensoriais humanas e, de modo especial, a acuidade visual. Com esse intuito, a amplificação de sinais, por exemplo, é uma técnica bastante difundida, pois permite que o operador observe no mostrador variações detectadas pelo hardware do instrumento que, de outra forma, seriam completamente imperceptíveis. Apesar de todos esses cuidados, falhas humanas podem ocorrer, seja no comando do instrumento, seja na leitura de suas escalas ou mostradores analógicos. Neste último caso, em particular, por maiores que sejam os cuidados tomados durante o ensaio, não há como eliminar os erros de interpolação visual, que apresentam claro comportamento tendencioso, independente do operador escolhido ou do número de repetições do experimento (TOMITA; HONDA, 1994).

Com a virtual obrigatoriedade de cumprimento das normas de qualidade da série ISO-9000, a demanda por ensaios metrológicos tem aumentado de forma exponencial, obrigando os laboratórios credenciados a automatizarem, tanto quanto possível, os ensaios e calibrações. O caso particular dos ensaios metrológicos baseados em imagens ilustra bem as dificuldades hoje enfrentadas por esses laboratórios. Por causa da instrumentação utilizada, tais ensaios são extremamente demorados. Apenas para citar um exemplo, a calibração de 21 pontos de uma microescala com 1 mm de comprimento, utilizando o método tradicional, consome cerca de 3,5 horas de trabalho de um técnico

experiente [1]. Esse tempo, à primeira vista exagerado, é justificado pela necessidade de se obedecer às recomendações descritas nos procedimentos do ensaio. Por se tratar de uma tarefa repetitiva, com elevada probabilidade de ocorrência de erro humano, o operador é condicionado a agir sempre de forma muito cautelosa, comparando e analisando mentalmente os dados durante o processo.

A substituição do operador humano por um sistema de visão computacional, além dos inegáveis benefícios econômicos, traz profundas implicações a todo o processo de medição. Diferentemente da abordagem tradicional, em que as ações do operador são completamente abstraídas, admitindo-se apenas que ele possua destreza para realizar as medições, os novos instrumentos baseados em visão computacional requerem que o conhecimento visual-metrológico do operador seja modelado, explicitado e integrado ao restante do processo. Do ponto de vista da teoria da medida, essa mudança de paradigma é definida como "uma incorporação de escalas mais fracas ao processo de medição" (BERKA, 1983), de modo que, além da escala racional (quantificação de grandezas), foco dos processos tradicionais de medição, a nova abordagem se utiliza também das escalas nominal (reconhecimento de padrões) e ordinal (ordenação e classificação de padrões). Portanto, os resultados das medições certamente irão diferir dependendo do modelo desenvolvido para descrever o processo de interpretação das imagens observadas pelo operador ou, em outras palavras, da precisão dos instrumentos utilizados para efetuar medições nas escalas nominal, ordinal e racional.

O objetivo central deste trabalho é o desenvolvimento e implementação de instrumentos metrológicos de visão computacional para automação de uma categoria de ensaios de calibração de padrões bidimensionais em que o operador realiza ações de

[1] Dado fornecido pelo Laboratório de Metrologia da Divisão de Mecânica e Eletricidade do IPT.

observação e monitoração de imagens opticamente amplificadas, bem como comando de instrumentos de deslocamento mecânico e de aquisição de dados. Embora a referida classe de ensaios abranja um grande número de calibrações de laboratórios de padrões secundários, a validade da abordagem proposta será demonstrada através da automação de dois ensaios típicos: calibração de microescalas lineares e de microtexturas de peneiras de medição.

Diferentemente do que ocorre no processo metrológico convencional, em que todas as medidas derivam de deslocamentos mecânicos (figura.1.3-a), neste trabalho apenas os fatores de escala (mm/*nº de pixels*) podem ser determinados com auxílio desse método. As demais medidas são sempre extraídas das próprias imagens pelo software de visão computacional, desenvolvido para substituir as ações do operador (observação, monitoração, decisão e comando) por algoritmos de processamento de imagens, reconhecimento de padrões e ajuste de parâmetros (figura.1.3-b).

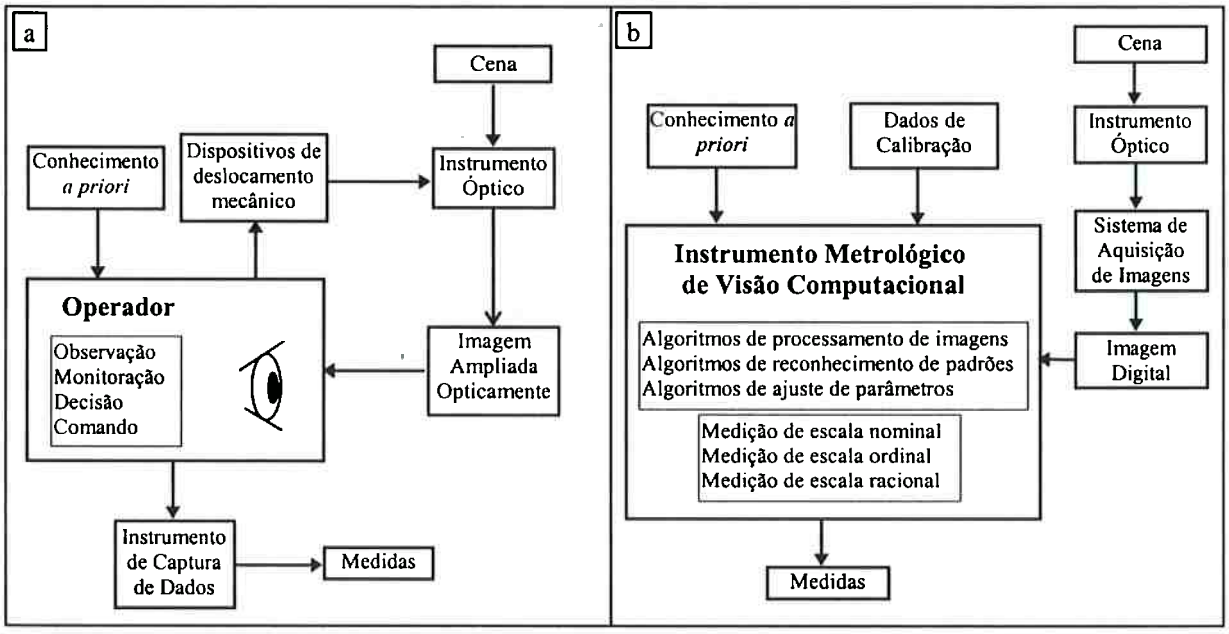

Figura 1.3: (a) Processo metrológico tradicional; (b) Instrumento metrológico de visão computacional

O desenvolvimento de instrumentos metrológicos de visão computacional para automação de ensaios baseados em imagens visa a atender simultaneamente três objetivos dos laboratórios de padrões secundários: melhoria da qualidade das medidas, minimização de erros humanos e aumento da produtividade laboratorial. A satisfação desses objetivos é, em última instância, o que justifica este trabalho. Por outro lado, buscando-se situá-lo no âmbito da literatura técnica recente, podem-se observar os seguintes aspectos inovadores que recomendam o seu desenvolvimento:

1. Aderência ao conceito de instrumentação inteligente: Conforme salientado em FILKENSTEIN (1994), os modernos instrumentos de medida, capazes de emular determinados sentidos e habilidades humanas, devem ser projetados combinando-se técnicas convencionais de instrumentação e medição com métodos inteligentes modelando problemas cognitivos tais como percepção, inteligência e raciocínio. O projeto de instrumentos metrológicos computacionais insere-se perfeitamente dentro desse escopo, onde a visão e perícia do metrologista constituem, neste caso, o alvo do processo de modelagem.
2. Uso das escalas fracas e de múltiplos paradigmas de medição: Na grande maioria das aplicações industriais de visão computacional, os algoritmos são projetados apenas para confirmar ou não a presença de uma determinada característica na imagem, sem que, para tanto, sejam enfatizados os métodos de medição. Mesmo quando realizam medições, estas não são devidamente analisadas ou destacadas. Neste trabalho, ao contrário, todas as etapas do processo são descritas como operações de medição, em várias escalas, que utilizam os diversos paradigmas de visão computacional como ferramentas alternativas de medição, da mesma forma como se se operasse com um paquímetro ou um relógio comparador. Tal abordagem tem o mérito de facilitar a

representação do conhecimento do domínio do problema e sua generalização para novas aplicações em problemas similares.

O desenvolvimento deste trabalho será feito na forma como segue. No capítulo 2 apresenta-se uma revisão bibliográfica da literatura, em que são analisadas algumas aplicações relevantes de visão computacional a problemas industriais de inspeção e metrologia. O capítulo 3 aborda o problema de medição baseada em imagens à luz de conceitos de visão computacional e da teoria da medida, buscando-se, através dessa aliança, desenvolver diferentes metodologias de medição aplicáveis à automação dos ensaios de calibração de microescalas e microtexturas de peneiras de medição. No capítulo 4, são apresentados os instrumentos metrológicos de visão computacional construídos em conformidade com as abordagens propostas no capítulo anterior. O capítulo 5 descreve os experimentos realizados com os referidos instrumentos e discute os resultados obtidos. No capítulo 6 apresentam-se as conclusões e sugestões para futuros trabalhos.

2 - REVISÃO BIBLIOGRÁFICA

Conforme já mencionado, o uso da visão computacional como ferramenta de medição de laboratórios metrológicos de padrões secundários é um tema pouco freqüente na literatura, o que em grande parte se deve à também pequena quantidade de trabalhos dedicados à análise das características de precisão dos sistemas de aquisição de imagens e dos algoritmos de visão computacional. Aliás, em 1987 HARALICK (1987) já afirmava haver um excessivo predomínio de trabalhos em visão computacional em que "a preocupação em fazer alguma coisa interessante era muito maior do que em caracterizar o desempenho dos algoritmos." Razão mais forte ainda para que a visão computacional venha se mantendo afastada do universo de problemas abordados pelos laboratórios metrológicos, tem sido a contínua e crescente demanda por aplicações industriais, especialmente nas áreas de inspeção visual automática, controle de qualidade, engenharia reversa e nanometrologia.

Por esse motivo, e para que algumas idéias importantes para a compreensão do trabalho possam ser introduzidas, a revisão bibliográfica será apresentada na forma como segue. Quanto às aplicações industriais, serão considerados apenas os trabalhos que tratam de medição/inspeção de objetos bidimensionais a partir de imagens desses objetos amplificadas por microscópio ou sistema de lentes e capturadas por câmera CCD e hardware de aquisição de imagens. Com isso ficam excluídos, por exemplo, tópicos como medição de objetos tridimensionais e medição baseada em imagens de microscopia eletrônica. Quanto às aplicações em laboratórios de padrões secundários, além de trabalhos sobre ferramentas metrológicas baseadas em visão computacional,

também serão discutidas algumas investigações a respeito das características de desempenho dessas ferramentas, com o intuito de aplicá-las em ensaios laboratoriais.

2.1 Visão Computacional como Ferramenta Metrológica Industrial

Em inspeção visual automática, os algoritmos de visão computacional operam como construtores de um instrumento de medição baseado nas escalas nominal e/ou ordinal. O primeiro caso corresponde à identificação de defeitos de naturezas distintas como, por exemplo, curto-circuito e circuito-aberto; o segundo caso, à identificação de defeitos de mesma natureza e diferentes níveis de gravidade, sendo um exemplo típico as variações dimensionais.

Um sistema muito simples de inspeção visual automática, desenvolvido por OKAWA *et* MIZUNO (1991), propõe-se a identificar certos tipos de defeitos de forma que ocorrem em *pellets* de diodos. Por causa de sua geometria em forma de tronco de cone, *pellets* perfeitos vistos de topo em microscópio apresentam-se como duas regiões circulares concêntricas nitidamente delimitadas por duas circunferências. Para que se possa utilizar esse padrão de referência no processo de inspeção, determina-se, para um conjunto de imagens "ideais", o centro médio único das regiões circulares e constrói-se um vetor de referência $\boldsymbol{R}$ com os valores médios dos comprimentos de n pares de arcos medidos ao longo das fronteiras das regiões em passos angulares de $360/n$ graus; o mesmo processo, aplicado a uma imagem-teste, produz o vetor $\boldsymbol{T}$. A inspeção é, então, realizada, comparando-se um a um cada um dos componentes T_i e R_i, do que resultam três possíveis resultados: *pellet* normal se, para $\forall\ i$, $T_i \approx R_i$; *pellet* defeituoso do tipo 1 (agregação de material), se $\exists\ i$ tal que $T_i > R_i$; *pellet* defeituoso do tipo 2 (remoção de material), se $\exists\ i$ tal que $T_i < R_i$. Esse método de inspeção é, portanto, um processo de

medição que utiliza um instrumento (vetor *R*) baseado em uma escala nominal com três possíveis valores: (*normal*, *defeito_tipo_1*, *defeito_tipo_2*).

SANZ *et* PETKOVIC (1988) desenvolveram um sistema de inspeção visual automática para identificação de duas grandes categorias de defeitos presentes em cabeçotes de discos magnéticos: material particulado aderido à superfície e deformações locais das bordas do cabeçote. Contrastando com a técnica adotada no trabalho anterior, que utilizava uma única imagem-referência para a construção do instrumento de medida, os autores deste trabalho identificam deformações locais utilizando um novo padrão de referência a cada inspeção. Tais padrões são obtidos aplicando-se a transformada de Hough (ILLINGWORTH; KITLER, 1988) na localização das fronteiras retilíneas da imagem do cabeçote. Graças à estreita variabilidade dimensional dos discos magnéticos, exceptuando-se os defeitos localizados não se observam alterações nas dimensões globais da imagem de diferentes cabeçotes. Em outras palavras, cada novo instrumento de medida assim construído tem precisão suficiente para efetuar medições em escala ordinal, identificando vários níveis de discrepância dimensional ao longo das bordas da imagem do cabeçote.

O sistema de inspeção visual automática de circuitos integrados desenvolvido por DOM *et al* (1988) destina-se, exclusivamente, à inspeção de estruturas repetidas (por exemplo, matrizes de memória) agrupadas em uma dada região de um *chip*. Graças a um sistema de posicionamento mecânico bastante preciso, capturam-se imagens em que os padrões retangulares repetidos ficam perfeitamente alinhados em relação aos seus eixos. Considerando-se que a imagem é periódica e que é muito improvável que defeitos exatamente iguais ocorram na mesma região ao longo de três períodos consecutivos, cada novo padrão é inspecionado, pixel a pixel, com referência aos padrões anterior e posterior. Um pixel é considerado "defeituoso" se o seu nível de cinza

diferir de modo significativo do nível de cinza dos dois pixels correspondentes nos padrões anterior e posterior. O aspecto interessante desse sistema é a utilização de um padrão de referência construído a partir de informações não corrompidas extraídas da própria imagem que está sendo medida.

KHALAJ *et al* (1994) também desenvolveram um sistema para inspeção visual automática de *chips* e demais dispositivos eletrônicos apresentando repetição de padrões. Semelhantemente ao trabalho anterior, neste, também, a redundância de informação armazenada em imagens de padrões repetidos é fator essencial para se construir uma escala de referência; neste caso, porém, a metodologia utilizada é muito mais complexa. Projetando-se a imagem nas direções horizontal e vertical, resultam dois perfis $P(x)$ e $P(y)$ descrevendo a variação média de níveis de cinza ao longo das duas direções. Por se tratarem de realizações de perfis periódicos ideais, com freqüências espaciais fundamentais f_{0x} e f_{0y}, estas são estimadas a partir de métodos de mínimos quadrados aplicados a medidas baseadas em $P(x)$ e $P(y)$. Finalmente, um padrão retangular de medição é construído arbitrando-se suas dimensões L_x, L_y e preenchendo-se cada um de seus pixels com os valores médios dos níveis de cinza dos pixels da imagem amostrados a intervalos horizontais $1/f_{0x}$ e verticais $1/f_{0y}$.

O sistema de inspeção visual automática de placas de circuito impresso desenvolvido por RODRIGUEZ *et* MANDEVILLE (1993), utiliza como instrumento de medição o desenho de engenharia da placa armazenado em um arquivo CAD. Por causa da complexidade geométrica dos circuitos e dos finos espaçamentos e espessuras, toda a dificuldade da inspeção se concentra em modelar o processo de superposição da imagem teste com o instrumento de medição (desenho). Esse "alinhamento" é executado em duas etapas. Utilizando como referência a imagem de uma placa de calibração com pequenos quadrados equiespaçados, aplica-se o processo de calibração de câmera

(GONZALEZ; WINTZ, 1987), do qual resulta a transformação que mapeia os pixels da imagem no sistema de coordenadas do desenho de engenharia. Todavia, por causa das imprecisões do sistema de posicionamento mecânico, cada vez que uma nova região da placa é inspecionada, introduzem-se erros que invalidam o mapeamento anterior. Essas distorções são corrigidas através da aplicação da técnica de enquadramento de imagens [1] (MASCARENHAS, 1990), ou seja: determinam-se os parâmetros da transformação-afim que minimizam um funcional que mede a *distância* entre um conjunto de objetos selecionados na imagem-referência e localizados na imagem-teste; esta transformação, aplicada à imagem-teste, efetua então a superposição desta imagem com o "gabarito de medição".

Conforme originalmente demonstrado por MANDEVILLE (1985), detecção de defeitos e verificação dimensional em placas de circuito impresso podem ser realizados por métodos outros que a simples comparação pixel a pixel. Nesse trabalho, operadores morfológicos clássicos (SERRA, 1989) são utilizados na construção de algoritmos para verificação de espessuras mínima e máxima de trilha, detecção de circuitos abertos, detecção de curto-circuitos, verificação de posição e geometria dos terminais. Através da aplicação de operadores morfológicos, a imagem original sofre simplificações drásticas, que acabam por transformá-la em um grafo. Com isso, o processo tradicional de medição também se altera radicalmente: a escala de referência de medida deixa de ser uma imagem ideal e passa a ser um grafo-ideal; a comparação deixa de ser entre imagens e passa a ser entre grafos. Contrastando com o método de comparação pixel a pixel, extremamente sensível a erros de alinhamento, essa outra abordagem é bem mais robusta, não exigindo o desenvolvimento de sistemas mecânicos complexos de posicionamento e algoritmos sofisticados de enquadramento de imagens.

---

[1] *Registration*, em inglês, inadequadamente traduzida para o português como *registro*.

A partir da abordagem sugerida por MANDEVILLE (1985), o desenvolvimento de sistemas de inspeção visual automática baseados em modelo ganhou ímpeto considerável, cabendo ao trabalho de DARWISH *et* JAIN (1988) um destaque especial. Esses autores implementaram uma base de conhecimento contendo algoritmos que utilizam operadores morfológicos para transformar a imagem da placa de circuito impresso em um conjunto de sub-grafos descritores dos circuitos, representando-os por meio de redes semânticas. Concluída essa transformação, a superposição dos sub-grafos das imagens teste e referência é formulada como um problema de otimização discreta, com espaço de busca previamente limitado graças à grande quantidade de informação agregada aos nós e ramos das redes semânticas. Portanto, um único instrumento de medição com a geometria complexa de toda placa, é substituído por diversos "gabaritos semânticos de medição", representando no conjunto exatamente a mesma geometria; ao invés de um complexo problema de monitoramento e superposição de imagens, o processo de medição é formulado como um problema de quebra-cabeças.

Apesar de a inspeção visual automática ser, indiscutivelmente, o principal campo de aplicação dos algoritmos de visão computacional, estes também têm sido utilizados em atividades de medição dimensional, cabendo destaque especial às aplicações em controle de qualidade e interfaceamento com sistemas CAD. Alguns exemplos de aplicações como essas são discutidos a seguir.

IRVING *et al* (1989) desenvolveram um sistema de medição de objetos planos e geração automática de suas representações CAD, para manutenção da coerência de informações de engenharia em ambientes de fabricação de peças em chapa metálica. Nesse sistema, os algoritmos de visão computacional são projetados para recuperarem informações geométricas de objetos de forma genérica, descritos por segmentos de reta, circunferências e arcos de circunferência. Após a detecção e afinamento das bordas da

imagem (BALLARD; BROWN, 1982), aplica-se a transformada de Hough — na versão original, para localização de retas, e na versão modificada, para localização de circunferências. Os limites de cada segmento de reta são localizados com auxílio da imagem limiarizada e os segmentos assim obtidos são emparelhados dois a dois segundo um critério de proximidade de vértices. Após o emparelhamento, ajustam-se arcos de circunferência tangentes aos segmentos e passando pelos pixels da imagem de bordas, completando-se assim o processo de medição e recuperação de forma do objeto.

BELAÏD (1990) desenvolveu um sistema protótipo para controle de qualidade dimensional de porcas de segurança em linha de produção, requerendo tolerâncias inferiores a 0,1 mm. Para tanto, foram implementados três sub-sistemas independentes de captura de imagens: uma câmera superior, de topo, para a observação dos diâmetros externo e interno da porca, uma câmera frontal para a observação do perfil da porca e uma câmera superior inclinada para a observação dos fios da rosca. A calibração das câmeras, para determinação dos fatores de escala horizontal e vertical, é realizada utilizando-se como referência imagens de uma porca com dimensões conhecidas. Para as imagens tomadas com a câmera de topo, após a aplicação de um algoritmo de detecção e afinamento de bordas, utilizam-se procedimentos de busca para a seleção dos pixels pertencentes aos contornos poligonal e circular. A medição dos diâmetros interno e externo da porca é realizada ajustando-se, pelo método dos mínimos quadrados, uma circunferência e um hexágono a cada um dos conjuntos correspondentes de pontos. Para as imagens capturadas com a câmera frontal, também se aplicam algoritmos de detecção e afinamento de bordas, porém o posterior ajuste de retas e curvas é mais complexo pois, para cada tipo de porca deve ser considerado um diferente perfil. A câmera superior inclinada é localizada em um ambiente escuro, de modo a apenas observar o reflexo de um feixe de laser incidente tangencialmente à rosca. A imagem capturada por

essa câmera apresenta-se como uma seqüência de objetos claros, dos quais somente os maiores correspondem a fios de rosca. Após eliminação dos ruídos com o auxílio de operadores morfológicos, a contagem dos objetos remanescentes fornece o número de fios da rosca.

Um sistema de visão computacional para medição em tempo real das dimensões características de roscas externas é apresentado por HUNSICKER *et al* (1994). Neste sistema, parte integrante de um ambiente de controle de qualidade de manufatura, cada novo fuso a ser inspecionado é agarrado por um dispositivo automático de fixação e mantido alinhado paralelamente ao eixo horizontal da câmera durante o tempo necessário para a medição. Os algoritmos de visão computacional são escritos para satisfazer à difícil combinação de requisitos envolvendo mínimo tempo de execução e elevada precisão das medidas. Por esse motivo, os pontos de referência para as medições são identificados em duas etapas: a posição inicial é obtida correlacionando-se a imagem com um dos padrões representando *início_de_rosca*, *perfil_de_dente* e *término_de_rosca*; a partir daquela posição, varrendo-se a imagem na direção de interesse com janelas unidimensionais de 3 pixels de largura, determinam-se as coordenadas das bordas da imagem com precisão de subpixel (STAHLBERG, 1995). Utilizando essa estratégia de busca, foram elaborados algoritmos para determinação dos valores mínimo, máximo e médio do comprimento total da rosca e de seus diâmetros interno e externo. Quanto ao diâmetro de passo, este é obtido através de um método iterativo que converge quando o seu valor corresponder, na imagem do perfil, à linha que separa as cristas e raízes dos dentes em dois conjuntos de objetos com mesma área.

DOIRON (1989) implementou um sistema protótipo de visão computacional para monitoração do desgaste da ferramenta de corte de um torno, com o proposito de melhorar a qualidade do processo de usinagem. Nesse sistema, as alterações de forma

detectadas na ferramenta são utilizadas para corrigir, em tempo real, as coordenadas de sua trajetória, descritas em programa de comando numérico. Acoplada à mesa *X-Y* do torno, uma plataforma de medição paralela ao plano da ferramenta captura imagens periodicamente observadas por uma câmera CCD acoplada a um microscópio de medição. Nessas imagens a ferramenta apresenta-se como um perfil composto por duas retas não ortogonais interligadas por um arco de circunferência *A*, porém no programa de comando numérico ela é definida como um objeto puntual *F* com coordenadas dadas pela intersecção de duas retas paralelas às direções *X, Y* e tangentes ao arco *A*. Portanto, o processo de monitoração requer localização precisa do ponto *F* na imagem e posterior descrição das coordenadas de *F* no sistema de referência *XY* do torno, no qual se baseiam as instruções de comando numérico. No plano-imagem, as coordenadas do ponto *F* são imediatamente determinadas após o ajuste de um arco de circunferência através das bordas da imagem, geradas por um detector de bordas com precisão de subpixel. Os parâmetros da transformação que mapeia as coordenadas da imagem no sistema de referência do torno são determinados através do processo de calibração da câmera, adotando-se como escala de medida um pequeno objeto circular montado sobre uma mesa cinemática, no mesmo plano da ferramenta.

2.2 Visão Computacional em Laboratórios Metrológicos de Padrões Secundários

Em laboratórios metrológicos de padrões secundários, a visão computacional tem sido mais objeto de análise do que de aplicação e, quando isso ocorre, em geral as medições dimensionais continuam a ser realizadas pelos tradicionais métodos baseados em deslocamento mecânico.

LASSILA *et al* (1994) aplicaram visão computacional na automação de ensaios de calibração de escalas graduadas com até 1m de comprimento. Tradicionalmente,

esses ensaios utilizam interferômetro de Michelson para medição de deslocamentos mecânicos e um microscópio fotoelétrico para detecção dos traços da escala; porém, quando este é deslocado dinâmica e automaticamente ao longo da escala, flutuações de velocidade acarretam imprecisão na detecção dos traços e, portanto, erros nas medidas. No sistema alternativo proposto, uma plataforma de visão computacional com microscópio óptico, câmera CCD e hardware de aquisição de imagens, substitui o microscópio fotoelétrico e, graças aos sinais de sincronismo de vídeo gerados pelo hardware de aquisição, é possível reproduzir as mesmas condições de captura de imagens para cada novo traço, tornando o processo de observação imune a flutuações de velocidade. A função de detecção desempenhada pelo microscópio fotoelétrico é facilmente emulada por algoritmos de visão computacional. Inicialmente, alinham-se os traços da imagem com o eixo horizontal da câmera e arbitra-se uma posição adequada para servir como "janela de observação". Dessa forma, a presença de cada novo traço da escala é indicada por uma súbita variação do nível médio de cinza da janela. Quando isto ocorre, registram-se as posições do motor de passo, dispara-se o interferômetro e aguardam-se as chegadas do início e fim do sinal de sincronismo de vídeo para se capturarem duas imagens do traço. Projetando-se cada uma dessas imagens na direção horizontal, obtém-se perfis cujos pontos de máximo correspondem às posições extremas da imagem do traço durante o intervalo de sincronismo. A posição média é então utilizada como referência para a contagem do número de franjas registradas pelo interferômetro.

Durante a implementação de uma estação de usinagem de alta precisão monitorada por um sistema de visão computacional, DOIRON (1990) aplicou aos componentes desse sistema uma série de testes que tinham por objetivo caracterizar e otimizar o seu desempenho metrológico. Capturando por longo período de tempo

imagens de uma pequena grade pré-calibrada, concluiu que o conjunto microscópio-câmera introduzia nas imagens distorção inferior a 0,1% e que as imagens capturadas apresentavam repetibilidade da ordem de 0,1 pixel. Dentre os diferentes esquemas de iluminação testados, o que menos corrompia as bordas da imagem era a iluminação difusa traseira. Detectores de borda convencionais foram também testados, porém apenas um detector especialmente projetado, que utilizava informações sobre as características médias de variação de níveis de cinza da imagem, foi capaz de detectar bordas com a precisão requerida pelos subsequentes processos de medição. Após aplicar uma série de testes a esse detector, obteve importantes conclusões acerca de seu comportamento relativamente a translação e rotação: invariante a translação, porém nitidamente sensível a rotações do objeto, apresentando máxima precisão quando as bordas estavam alinhadas na direção vertical da imagem e mínima, quando na direção horizontal.

JIANG *et* SHIAU (1990) desenvolveram uma metodologia baseada em projeto de experimentos para avaliar as características metrológicas de desempenho de sistemas de visão computacional dedicados a inspeção e medição industriais. Nessas atividades, as variáveis relevantes para o desempenho do sistema — características do sensor, fonte de iluminação, reflectividade e orientação da superfície do objeto observado — podiam ser controladas através de 7 fatores, cada um deles apresentando 3 níveis: distância focal da lente (25; 50; 75 mm); cor de fundo (branco; amarelo; verde); distância $D_{o\text{-}o}$ entre dois objetos (2; 4; 6"); distância $D_{c\text{-}o}$ entre câmera e objeto (6,6; 8,2; 9,8'); filtro analógico (nenhum; passa-baixa; mediana); iluminação (luz fluorescente normal; luz fluorescente de alta freqüência; fibra óptica); ângulo entre o eixo da câmera e a superfície do objeto (60°; 75°; 90°). Utilizando-se o modelo de experimentos fatorial-

fracionados e calculando-se, para cada um deles, a exatidão [2] e repetibilidade das medidas da distância entre dois objetos observados na imagem, pôde-se identificar os fatores importantes para o desempenho do sistema. Com relação à exatidão, eram relevantes a distância $D_{o\text{-}o}$ entre objetos e a interação entre $D_{o\text{-}o}$ e a distância $D_{c\text{-}o}$ entre câmera e objeto; com relação à repetibilidade, eram significativos os fatores cor de fundo e a distância $D_{c\text{-}o}$. A partir desses resultados preliminares, foram construídas tabelas que permitiam identificar os níveis ideais dos diversos fatores que otimizavam simultaneamente a exatidão e a repetibilidade das medições.

[2] O termo *exatidão* é aqui utilizado com o mesmo significado que "*accuracy*".

3 - MEDIÇÃO BASEADA EM IMAGENS

Conforme ilustrado na figura 1.3, a construção de um instrumento metrológico de visão computacional requer a substituição de um conjunto de conhecimentos e habilidades sensoriais humanas por uma seqüência de operações de processamento de imagens, reconhecimento de padrões e ajuste de parâmetros, de forma tal que o instrumento assim constituído produza medidas cuja exatidão e repetibilidade sejam compatíveis com as do método convencional.

GNIOTEK (1997) sintetiza esse processo de construção do novo instrumento metrológico como a "realização de medições *criativas* para a geração de um modelo de medições *rotineiras*", ou seja: na fase de construção do instrumento realizam-se medições para identificar a estrutura do modelo que melhor descreve o conhecimento metrológico focalizado, enquanto que na fase operacional as medições se limitam à identificação dos parâmetros do modelo cuja estrutura já fora previamente determinada. Ao abordar essa mesma categoria de medições ditas *criativas*, FERRIS (1997) descreve-as como processos de mapeamento e aprendizagem das características e atributos de uma *realidade* observada através de experimentos, em que: 1º) *a priori,* essa *realidade* manifesta-se como um conjunto de conceitos cujos atributos, inter-relações e relevâncias são conhecidos de forma parcial ou vaga; 2º) ferramentas de medição são desenvolvidas e testadas com o intuito de identificar as características da *realidade* observada; 3º) existe uma escala para expressar os resultados das medições; 4º) existe um conjunto de possíveis conclusões acerca da *realidade*; 5º) conclusões são obtidas a partir de transformações aplicadas às medidas realizadas.

A abordagem dos processos de medição *criativa* proposta por FERRIS (1997), pela sua generalidade e capacidade de síntese, é bastante adequada ao planejamento da construção de instrumentos metrológicos de visão computacional. Dessa forma, para cada um dos ítens acima referidos, serão estabelecidas correspondências com os aspectos pragmáticos do problema proposto, conforme sinteticamente representado no diagrama da figura 3.1.

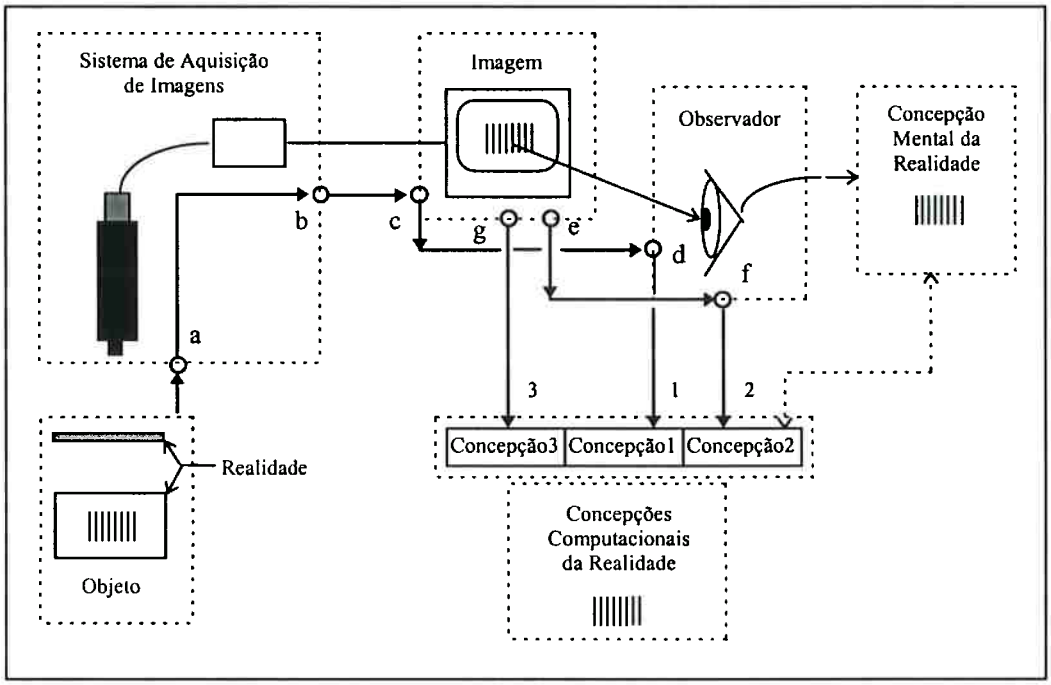

Figura 3.1: As várias concepções computacionais da realidade.

Ao se automatizar um processo de medição baseado em imagens utilizando métodos de visão computacional, a *realidade* que se procura identificar é, em última instância, a geometria do objeto a ser medido. Todavia, essa *realidade* é observada através de duas grandes barreiras de filtros, o que faz com que parte de seu conteúdo permaneça velado. A primeira dessas barreiras é o conjunto {*microscópio de medição, câmera CCD, iluminação, hardware de aquisição*}, que transforma o objeto original em uma imagem; a segunda é o operador humano que, ao observar a imagem no monitor de

vídeo, constrói mentalmente um modelo do objeto real. Assim é que, dependendo da exatidão requerida para o instrumento metrológico, diversas *concepções computacionais da realidade* podem ser consideradas. Um experimento de medição bastante exato consideraria a identificação de parâmetros, processos e atributos de uma *concepção*$_1$ abrangendo o sistema de aquisição de imagens e o conhecimento visual-metrológico do operador (circuito *a-b-c-d-1*). Em se buscando fidelidade ao processo clássico de medição, ignorar-se-ia a presença do sistema de aquisição de imagens, uma vez que o próprio operador humano "confia" nas imagens observadas; em tal situação, a *concepção*$_2$ a ser identificada abrangeria apenas o conhecimento visual metrológico do operador (circuito *e-f-2*). Em uma situação mais restritiva ainda, tanto o sistema de aquisição de imagens quanto o operador seriam ignorados, e a *concepção*$_3$ se limitaria apenas à imagem do objeto (circuito *g-3*).

As ferramentas a serem utilizadas na construção dos instrumentos metrológicos dependem da particular *concepção computacional da realidade* focalizada. Considere-se, inicialmente, a medição no âmbito da *concepção*$_3$: tem-se uma imagem que representa, por exemplo, uma escala linear (conjunto de traços quase equiespaçados). Nesta situação, a ferramenta de medição deverá se limitar a usar apenas informações da própria imagem para identificar a distância média entre traços, o que pode ser feito, por exemplo, com o auxílio de métodos de estimação de freqüências espaciais (KHALAJ *et al*, 1994). No âmbito da *concepção*$_2$, a construção de um modelo emulando o conhecimento visual-metrológico do operador possibilita utilizar variado número de ferramentas de visão computacional. Adotando-se, por exemplo, o processo evolutivo de identificação de conhecimento (MARI, 1996) como referência para a extração do conhecimento visual humano, aplicam-se à imagem, nesta seqüência, operações de medição nas escalas nominal, ordinal e racional, acumulando-se evidências oriundas das

escalas mais fracas que possibilitam o gradual refinamento de conhecimento à medida que se migra para as medições em escalas mais fortes. Em visão computacional, esse mesmo processo é designado pela expressão *"from pixels to pictures"*, correspondendo a uma seqüência de operações de aprimoramento de imagens, segmentação de imagens e identificação de contornos, tendo por objetivo promover a informação de baixo nível (*pixels*) a uma representação gráfica altamente estruturada (*pictures*) que possibilite inferências e medições quantitativas. Habilidades metrológicas do operador podem também ser emuladas por ferramentas de visão computacional: por exemplo, algoritmos de enquadramento de imagens e certos algoritmos de segmentação baseados em operadores morfológicos emulam a monitoração do alinhamento de um objeto-referência com o objeto a ser medido. No âmbito da $concepção_1$, além de todos os recursos pertinentes à $concepção_2$, devem-se incluir recursos capazes de minimizar o impacto da degradação da imagem através dos componentes do sistema de aquisição sobre os subseqüentes algoritmos de medição.

Os resultados finais das medições realizadas com os instrumentos metrológicos de visão computacional devem ser medidas quantitativas, em escala racional. Lembrando que medições em imagens, nessa escala, produzem resultados expressos em número de pixels, é fundamental que aqueles instrumentos disponham de um método de calibração para converter as coordenadas dos pixels da câmera em unidades métricas.

Em relação à construção de instrumentos metrológicos de visão computacional, o processo de medição *criativa* oferece um conjunto bastante extenso de possíveis conclusões. Para cada uma das *concepções computacionais da realidade* adotadas pode-se concluir, por exemplo, que, das n ferramentas disponíveis, as melhores medidas da *realidade* se obtêm com o conjunto $\{F_i,F_j,F_k\}$, sendo que o desempenho da ferramenta F_i, com m parâmetros de entrada $(p_1,p_2,...p_m)$, é melhor quando $(p_1=a_1, p_2=a_2,..., p_m=a_m)$.

Tais conclusões são obtidas quando se utilizam métricas indicadoras da qualidade das medições efetuadas com cada um dos instrumentos, adotando-se múltiplas *concepções computacionais da realidade* e combinações variadas de ferramentas e parâmetros. Por exemplo, pode-se tomar como referência um ensaio convencional e utilizar como métrica de qualidade a média quadrática da diferença entre os resultados desse ensaio e as medidas geradas por um dado instrumento metrológico de visão computacional. Desde que a resolução deste instrumento seja inferior à do instrumental usado no ensaio convencional, tal métrica possibilita ajustar gradualmente ferramentas e parâmetros, de modo que se componha um instrumento com o melhor desempenho possível.

Nos próximos ítens serão apresentados alguns conceitos, ferramentas e recursos de visão computacional que terão papel relevante na construção de instrumentos metrológicos para automação de dois ensaios clássicos realizados em laboratórios de metrologia de padrões secundários — calibração de micro-escalas e de microtexturas de peneiras de medição. No âmbito exclusivo da $concepção_1$, a discussão sobre os processos de formação e captura de imagens terá por objetivo o desenvolvimento de métodos e ferramentas que minimizem os efeitos do processo de degradação da imagem sobre os subsequentes algoritmos de medição. No âmbito das *concepções 1* e *2* serão apresentados algoritmos de realce de bordas, limiarização, rotulação, localização e ajuste de retas, para a composição de instrumentos de medição baseados no processo evolutivo; além disso, serão também descritos algoritmos de morfologia matemática capazes de emular o processo de medição local por meio de gabaritos. Finalmente, para a construção de ferramenta de medição no âmbito da $concepção_3$, serão apresentados algoritmos que localizam retas ou padrões espaciais repetitivos na imagem, mas não buscam reproduzir quaisquer habilidades tipicamente humanas.

3.1 Formação, Degradação e Captura de Imagens

Considere-se o típico sistema de aquisição de imagens esquematizado na figura 3.2-a, composto por microscópio de medição, dispositivos de iluminação, câmera CCD e hardware de aquisição. Situando-se sobre a platina do microscópio um objeto plano com superfície refletora iluminada superiormente, a qualidade da imagem formada na saída do instrumental irá depender da interação *fluxo_de_luz_incidente→ superfície _do_objeto* e das características optoeletrônicas do sistema de aquisição de imagens.

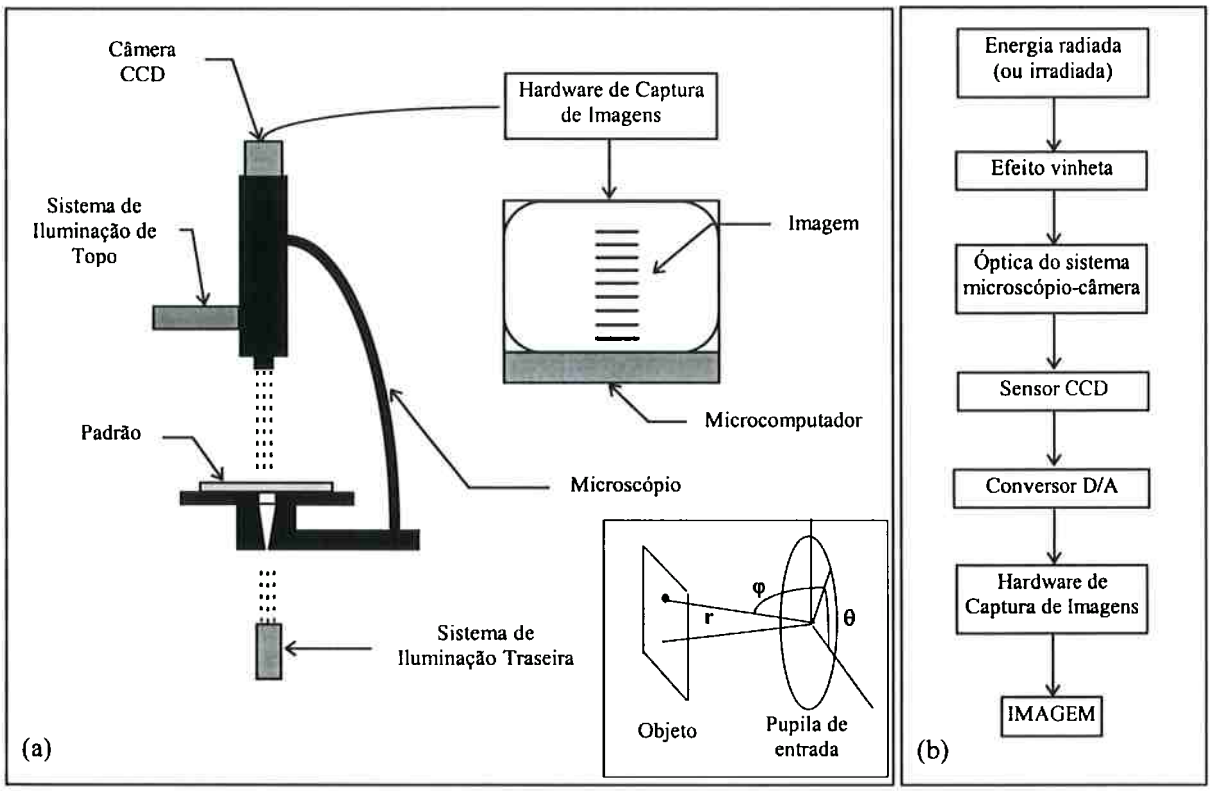

Figura 3.2: (a) Sistema de aquisição de imagens; (b) Etapas da transformação da energia radiada.

A interação entre as fontes de luz e a superfície do objeto observado determina a formação de um mapa de reflectância diretamente relacionado com as intensidades luminosas dos pixels da imagem. Conforme frisado na literatura (EL-HAKIM, 1990; LOTUFO, 1996), aplicações de visão computacional em inspeção visual automática e metrologia requerem rigoroso controle das condições de iluminação. Como as posições

das bordas da imagem dependem dos seus níveis de intensidade luminosa, iluminação não paralela e não homogênea cria falsos artefatos e modifica as posições e orientações dos contornos, tornando as imagens resultantes completamente intratáveis pelos algoritmos de medição.

A incidência dos raios luminosos sobre a superfície do objeto produz energia luminosa radiada [1] que, ao atravessar o caminho optoeletrônico do sistema, é submetida a uma série de transformações fotométricas, geométricas, ópticas e eletrônicas, as quais também acarretam degradação das informações originais do objeto. Acompanhando o diagrama de blocos da figura 3.2b, essas transformações, matematicamente descritas em (SUBBARAO; LU, 1994), são apresentadas a seguir.

Ao longo do caminho óptico constituído pelas lentes, aberturas e diafragmas do sistema microscópio-câmera, o feixe de luz cruzando a pupila de entrada do microscópio com informações fotométricas $f(r, \theta, \varphi)$ do objeto mapeáveis em uma *imagem focalizada ideal* $f(x,y)$, pode eventualmente sofrer vinhetamento. Neste caso, parte do cone de energia radiada pelos pontos do objeto mais afastados do eixo óptico do sistema é interrompido pelos diafragmas de campo, tornando as imagens desses pontos menos luminosas, ou seja, introduzindo não-uniformidade luminosa na imagem do objeto.

Ainda que fosse possível construir sistemas de lentes sem quaisquer aberrações, a resolução de um instrumento óptico seria limitada pelo fenômeno de difração de campo distante, que transforma cada ponto-objeto não em um ponto-imagem, mas no conhecido padrão de Airy — disco central claro circundado por anéis gradualmente mais escuros; aberrações nas lentes eventualmente afetam a simetria desse padrão ou redistribuem a energia luminosa entre o disco central e os anéis (KLEIN; FURTAK,

[1] No caso de iluminação traseira de objetos opacos, trata-se da energia irradiada pelo sistema de iluminação, que é transformada em uma imagem descrevendo a sombra do objeto observado.

1986). Assim é, que, a imagem resultante da transformação de um ponto-objeto pelo sistema óptico caracteriza a chamada *função de espalhamento de ponto* do sistema (*PSF(x,y)* [2]) que, para o caso de iluminação incoerente, pode ser considerada linear e espacialmente invariante. Nessas circunstâncias, o efeito do sistema óptico sobre a *imagem focalizada ideal f(x,y)* do objeto pode ser assimilado a uma operação de convolução entre *f(x,y)* e *PSF(x,y)*. No caso de o conjunto de lentes apresentar aberração do tipo *distorção*, os pontos da *imagem focalizada ideal* sofrerão, além disso, deformação geométrica, normalmente representada pela transformação de coordenadas:

$$x_d = \frac{2 \cdot x}{1 + \sqrt{1 - 4 \cdot k_r \cdot r^2}} \qquad y_d = \frac{2 \cdot y}{1 + \sqrt{1 - 4 \cdot k_r \cdot r^2}} \qquad r^2 = x^2 + y^2 \tag{3-1}$$

onde (x,y) é o ponto da *imagem ideal*, (x_d,y_d) é o ponto correspondente da imagem distorcida radialmente e k_r é o coeficiente de distorção radial (LENZ; FRITSCH, 1990).

Ao cruzar a pupila de saída do sistema óptico da câmera, o feixe de luz incide sobre o CCD, que efetua quantização escalar e amostragem espacial do sinal luminoso bidimensional contendo informações da *imagem focalizada ideal* atenuadas pelo efeito vinheta, filtradas pela função de espalhamento de ponto do sistema e quiçá distorcidas geometricamente por alguma transformação da forma 3-1.

O processo de quantização dos valores contínuos da intensidade luminosa da *imagem focalizada ideal* $f(x,y)$ em n níveis discretos de cinza da função quantizada $\hat{f}(x,y)$ é sempre realizado de forma a minimizar o erro quadrático médio $\mathcal{E}$ da diferença entre essas duas funções, ou seja, $\mathcal{E} = E\{(f - \hat{f})^2\}$. Conforme apresentado em (PRATT, 1978), esse erro é dependente do número de níveis de cinza n e da sua função densidade de probabilidade. Sendo n suficientemente grande e considerando-se os vários níveis de cinza da imagem equiprováveis, o erro de quantização esperado é $\mathcal{E} = 1/12n^2$.

[2] PSF, de *point spread function*.

Graças à excepcional qualidade dos processos litográficos utilizados para a fabricação de CCDs, são muito pequenas as deformações geométricas da *imagem ideal* devidas a variações posicionais dos sensores ao longo das linhas e colunas da matriz (inferiores a 0,01 pixel, segundo BEYER (1990)). Por outro lado, conforme descrito em (FLORY, 1985), durante o processo de digitalização, a informação espacial original é submetida a duas formas de degradação — uma é devida à arquitetura de transferência de cargas do CCD, a outra depende da relação entre as larguras da banda de freqüências do sinal bidimensional e os espaçamentos entre linhas e colunas da matriz de sensores.

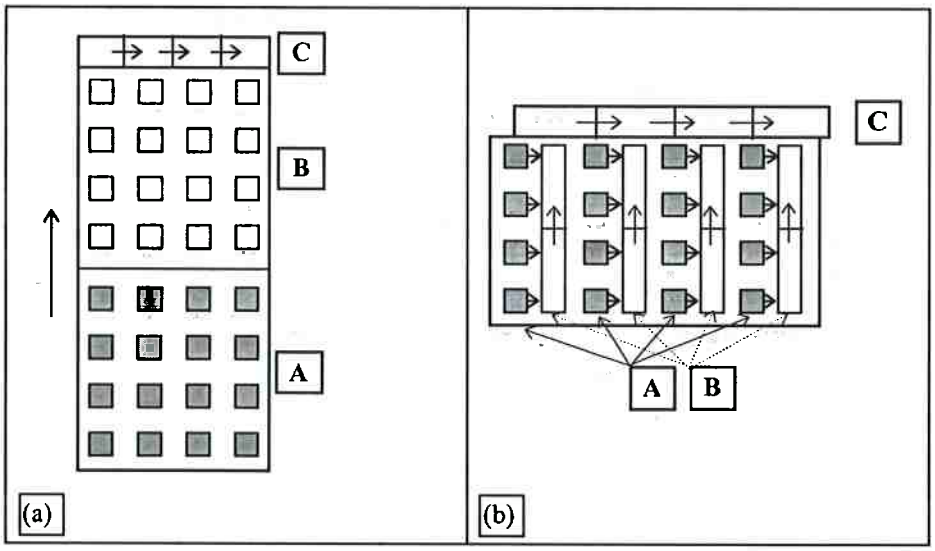

Figura 3.3: (a) Transferência quadro a quadro; (b) Transferência entrelaçada de linhas.

No método de transferência quadro a quadro (figura 3.3-a), o CCD apresenta, além dos registradores *C* de saída, uma matriz simétrica *A* de sensores e uma área de armazenamento *B*; após a integração temporal do sinal espacial, a matriz *A* é transferida para a área *B* que é então lida, linha a linha, pelos registradores *C*. Já o método de transferência entrelaçada de linhas (figura 3.3-b) apresenta os registradores de leitura *C* e uma matriz *A* assimétrica de sensores adjacentes, coluna a coluna, a um conjunto de registradores *B*, os quais são compartilhados alternadamente, em tempos distintos, por

sensores de duas linhas contíguas, ímpar e par. Em ambos os casos existem, ao longo das linhas e colunas do CCD, áreas insensíveis separando os sensores, que impedem que as informações da *imagem ideal* transportadas pela energia luminosa incidente nessas áreas sejam coletadas. Ademais, neste último método, a presença dos registradores *B* entre colunas aumenta a separação dos sensores nessa direção, o que introduz na imagem uma característica anisotrópica.

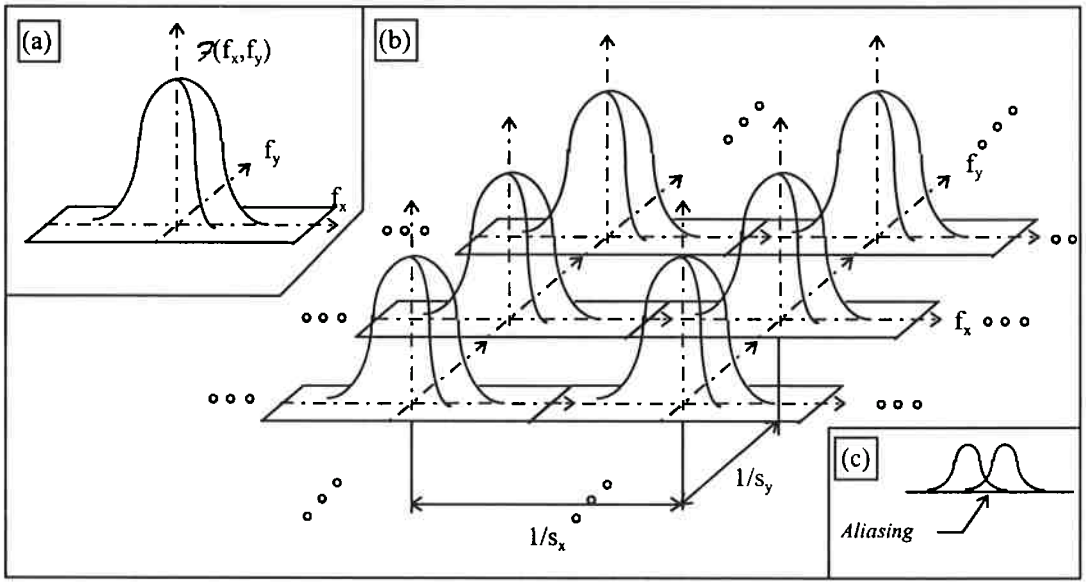

Figura 3.4: (a) Espectro da imagem original contínua; (b) Espectro da imagem digital; (c) Aliasing.

O processo de amostragem da função contínua bidimensional *f(x,y)* por um CCD com n_x colunas e n_y linhas de sensores de dimensões L_x , L_y separados por distâncias s_x, s_y, respectivamente, corresponde, no domínio das freqüências, à multiplicação de seu espectro de potência $\mathcal{F}(f_x , f_y)$ por um filtro retangular passa-baixa com freqüências espaciais de corte L_x^{-1} e L_y^{-1} seguida da convolução com uma função *pente de Dirac* bidimensional de freqüências espaciais s_x^{-1} e s_y^{-1} (figura 3.4). No espectro da imagem digital resultante, as bandas do espectro original replicado não podem se superpor, pois em tal caso a confusão (*aliasing*) entre as suas baixas e altas freqüências daria origem a

artefatos espúrios após o processo de reconstrução da imagem analógica. Para que isso não ocorra, as condições expressas pelo teorema de Nyquist devem ser satisfeitas, ou seja: $s_x^{-1} \geq 2f_{xmax}$ e $s_y^{-1} \geq 2f_{ymax}$, onde f_{xmax} e f_{ymax} são, respectivamente, as maiores freqüências espaciais f_x e f_y do espectro de potência original. Como, em geral, os inevitáveis ruídos de fundo da imagem alargam o seu espectro, favorecendo a ocorrência de *aliasing*, os processos de filtragem aplicados à *imagem focalizada ideal* pelo sistema óptico e pelos sensores do CCD, a despeito de seu efeito desfocalizante indesejável, acabam por trazer algum benefício, na medida que operam como filtros *antialiasing* (MARION, 1991).

Apesar de o processo de digitalização da imagem ser completamente realizado pelo CCD da câmera, estas são concebidas para operarem em conjunto com dispositivos de saída que utilizam como entrada sinais de vídeo padronizados (tipicamente, RS-170 ou CCIR). Na saída da câmera, portanto, um conversor D/A transforma a imagem digitalizada pelo CCD em um sinal unidimensional de vídeo composto, contendo tanto as informações das linhas da imagem quanto os sinais de sincronismo horizontal e vertical. Neste processo, o espectro da imagem digital é pré-filtrado com um filtro passa-baixa não ideal (figura 3.5) que, atenuando as suas frequências mais altas, reduz de forma significativa possíveis efeitos de *aliasing*. Em seguida, a imagem digital assim filtrada é convertida em um sinal analógico por um circuito eletrônico de interpolação. Sendo fisicamente irrealizável o processo de reconstrução ideal através da convolução da imagem digital com a função $sinc(\pi x f_{xmax}) \times sinc(\pi y f_{ymax})$, funções interpoladoras alternativas são utilizadas. No caso específico dos conversores D/A das câmeras CCD, SUBBARAO *et* LU (1994) salientam o predomínio de circuitos de interpolação de ordem 0 ou, eventualmente, de ordem 1. Tais esquemas primitivos de interpolação

introduzem erros consideráveis de reconstrução; no caso, por exemplo de interpolação de ordem 1, o erro de reconstrução estimado é de cerca de 3,7% (MARION, 1991).

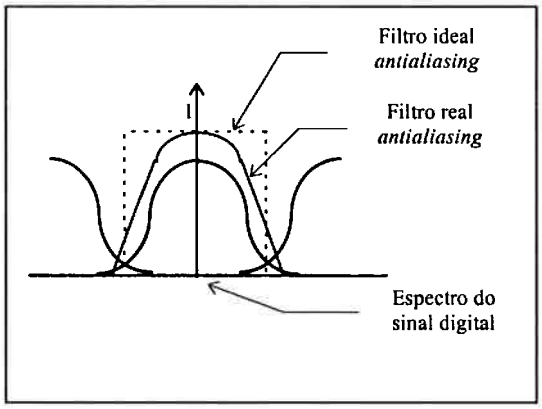

Figura 3.5: Aplicação de filtro anti-aliasing ao espectro da imagem digital

A imagem reconstruída e transmitida pela câmera CCD na forma de sinal de vídeo composto é então recebida pelo hardware de aquisição de imagens. Após ser condicionado por um filtro passa-baixa com freqüência de corte ajustável, o sinal de vídeo composto é tratado pelo *circuito separador de sinais de sincronismo*, que identifica esses sinais e os utiliza em seguida para isolar e digitalizar o sinal de imagem. Neste processo, dependendo do método utilizado para a sincronização entre os circuitos da câmera e do hardware de aquisição, podem ocorrer duas formas de degradação geométrica da imagem, bastante discutidas na literatura (LENZ; FRITSCH, 1990): a introdução de um fator de escala horizontal constante, tornando os pixels ligeiramente mais largos nessa direção; e o surgimento de oscilações (*jitter*) na posição de início de cada nova linha da imagem.

Após a aquisição, a imagem digital é transferida para a memória do computador, ficando então disponível tanto para processamento como para observação. Para possibilitar esta última aplicação, é necessário ainda recorrer ao sistema de I/O do computador que, através de seu circuito de conversão D/A, reconstrói a imagem analógica e apresenta-a no monitor de vídeo. Esta imagem final, a despeito de todo o

processo de degradação anteriormente descrito, é vista e interpretada pelo operador como uma representação fiel do objeto amplificado pela instrumentação óptica, sendo, por conseguinte, utilizada como referência para as subsequentes operações de medição.

Embora o processo neurofisiológico, que produz interpretações e conceitos sobre cenas ou imagens observadas, seja ainda pouco compreendido, alguns de seus aspectos de particular interesse para a metrologia têm sido identificados através do estudo da anatomia da retina e de experimentos psicofísicos realizados para medição da acuidade visual humana. Conforme descrito na literatura (PRATT, 1978; KROTKOV, 1986; YOUNG, 1995), a visão diurna é o resultado do processamento, no córtex cerebral, dos sinais elétricos gerados por uma rede discreta de células fotossensíveis (cones) que efetuam a amostragem de uma *imagem focalizada ideal* filtrada pela função de espalhamento de ponto do sistema ocular (córnea+pupila+cristalino). Essa rede de sensores, com espaçamentos inferiores a 10μm na região central da retina, impõe um limite físico à resolução espacial do sistema visual humano, comprovado por experimentos de medição de acuidade visual que indicam ser de 1 minuto o menor ângulo de separação entre dois objetos-teste que os tornam visualmente distinguíveis. Por outro lado, outros experimentos psicofísicos revelam também que o sistema visual humano tem capacidade de localizar a posição de objetos com exatidão muito maior do que a permitida, em princípio, pela sua resolução espacial.

KROTKOV (1986) afirma que a melhor explicação que os neurofisiologistas têm para esse fenômeno de hiper-acuidade visual baseia-se na *Teoria do Sinal Médio Local*, segundo a qual a posição de um objeto é obtida com maior exatidão porque é a resultante da combinação de informações espaciais locais com menor resolução, oriundas dos fotorreceptores estimulados pela imagem. Contudo, de acordo ainda com o mesmo autor, esse fenômeno poderia ser igualmente explicado se se admitisse que o

córtex cerebral humano aplica aos sinais elétricos provenientes da rede de fotorreceptores algum processo de interpolação, reconstruindo a imagem analógica original a partir desses dados digitais. Estas duas abordagens têm direta correspondência com as *concepções computacionais 1* e *2* apresentadas anteriormente. O ponto de vista de KROTKOV, que será explorado no próximo ítem (3.2), insere-se em um esquema em que se busca melhorar o desempenho dos algoritmos de medição através de filtros de reconstrução que ampliam a resolução espacial da imagem degradada pelo sistema de aquisição. Já a *Teoria do Sinal Médio Local* é a justificativa para a construção de diversos algoritmos de visão computacional com exatidão de subpixel que se aplicam diretamente à imagem digital, considerada representação fiel do objeto observado pelo sistema opto-eletrônico; tais algoritmos serão discutidos durante a exposição dos métodos de medição baseados no processo evolutivo de extração do conhecimento visual humano.

3.2 Minimização dos Efeitos da Degradação da Imagem

Conforme apresentado anteriormente, através dos vários componentes do sistema de aquisição de imagens, a *imagem ideal* do objeto sofre diversas formas de degradação fotométrica, geométrica e espectral. As primeiras são causadas por deficiências do sistema de iluminação, efeitos de vinhetamento ao longo do caminho óptico e limitações na sensibilidade do CCD. Degradações geométricas se devem a iluminação não paralela, distorção nas lentes e presença de *jitter* no circuito de digitalização do sinal de vídeo composto. As degradações do espectro de potência da imagem são causadas por uma combinação de operações de filtragem e reconstrução realizadas pelas lentes do sistema microscópio-câmera, sensores do CCD e circuitos D/A e A/D da câmera e do hardware de aquisição.

3.2.1 Minimização dos Efeitos das Degradações Fotométricas

Restrições impostas pela geometria do experimento determinam muitas vezes a construção de sistemas de iluminação que produzem campos de observação com intensidade luminosa variável. Defeitos desse tipo podem ser corrigidos multiplicando-se o nível de cinza de cada pixel da imagem por um fator de correção apropriado, obtido de forma experimental. RODRIGUEZ *et al* (1991), por exemplo, obtêm esses fatores utilizando os valores inversos dos níveis de cinza da imagem de uma superfície plana espelhada.

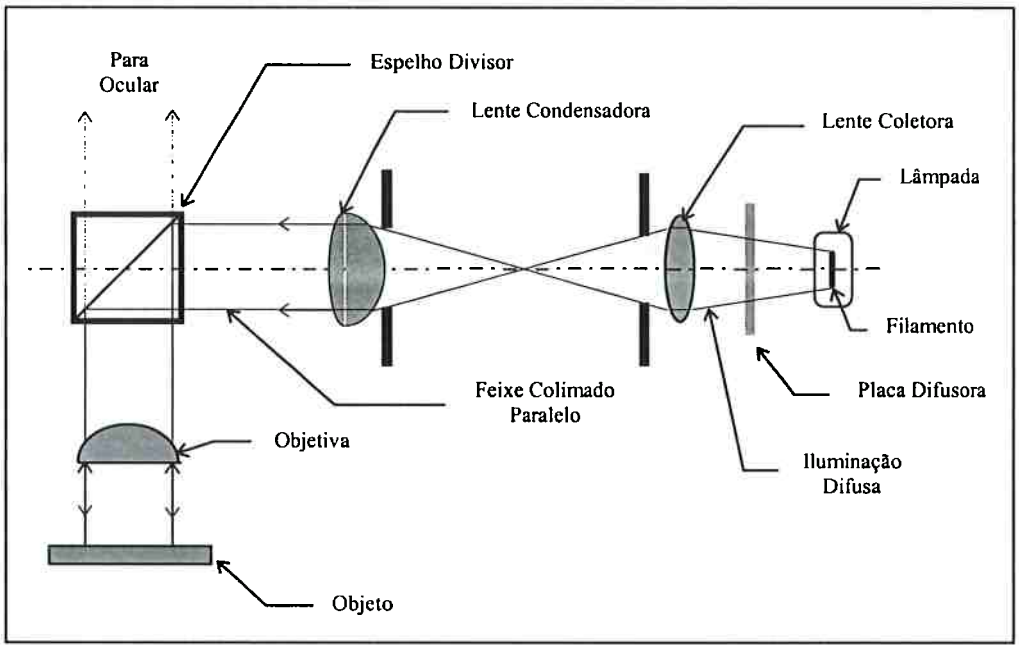

Figura 3.6: Sistema típico de iluminação usado em microscópios de medição.

Em microscópios de medição, felizmente, tanto o sistema de iluminação quanto o caminho óptico são concebidos para que raios luminosos paralelos projetem imagens uniformemente iluminadas do objeto observado. Para tanto, o vinhetamento é, em geral, eliminado através da inserção de lentes de campo em pontos estratégicos do caminho óptico (KLEIN; FURTAK, 1986). Quanto ao sistema de iluminação, a solução esquematizada na figura 3.6 é bastante típica: Uma placa de vidro não-polido situada

após a fonte luminosa difunde a luz, que é em seguida colimada por um conjunto de lentes coletora e condensadora em direção a um espelho divisor; o feixe paralelo aí incidente, após iluminar a superfície do objeto, retorna em direção à ocular do microscópio, produzindo imagem homogeneamente iluminada e não distorcida.

A degradação fotométrica da imagem devida à existência de áreas inativas na matriz do CCD pode ser minimizada utilizando-se a arquitetura de transferência de cargas quadro a quadro. Experimentos realizados por LENZ *et* FRITSCH (1990) revelam que a percentagem de área sensível nos CCDs baseados neste método é superior a 90%, porém no caso da transferência entrelaçada de linhas, pode ser inferior a 20%.

3.2.2 Minimização dos Efeitos das Degradações Geométricas

A identificação dos parâmetros que caracterizam a distorção das lentes das câmeras é um tema quase que onipresente na literatura sobre medição baseada em imagens. É fundamental salientar, porém, que a importância desse tema não se estende a todas as aplicações metrológicas, mas principalmente àquelas cuja instrumentação e métodos são originários das clássicas técnicas de fotogrametria de campo próximo (BROWN, 1971), posteriormente adaptadas para a medição de objetos tridimensionais. Ora, no caso que está sendo considerado, os objetos são observados não através de lentes objetivas de câmeras fotográficas, reconhecidamente imprecisas do ponto de vista metrológico, mas através de um microscópio de medição diretamente acoplado a uma câmera CCD. Conforme descrito em (KINGSLAKE, 1978), no projeto de instrumentos ópticos de qualidade, métodos de otimização permitem identificar os valores das variáveis do sistema de lentes (curvaturas, espaçamentos e espessuras) que minimizam a soma dos resíduos quadráticos $\Phi = R_1^2 + R_2^2 + \ldots + R_n^2$ das n aberrações Ab_i presentes no sistema. Ademais, como em geral sempre existe uma aberração Ab_i muito mais crítica

para a aplicação pretendida, é prática comum impor como condição de contorno do problema de minimização $R_i=0$. No caso específico de microscópios de medição, a presença de distorções geométricas na imagem inviabilizaria o processo de medição, razão pela qual nesses instrumentos esse tipo de aberração é virtualmente eliminado.

Conforme exposto em (BEYER, 1990), dependendo do processo de sincronização utilizado, diferentes níveis de *jitter* poderão afetar as características geométricas da imagem. A utilização de uma freqüência fixa de amostragem, mesmo que seja muito maior que a freqüência do sinal originalmente digitalizado pelo CCD, acarreta acúmulo de erros de sincronismo ao longo das sucessivas linhas, de modo que é esperado *jitter* de no mínimo 0,1 pixel entre a primeira e última linhas da imagem. Mais adequado é utilizar um circuito PLL (*phase locked loop*) implementando um método de correlação cruzada que utiliza um sinal de referência $hsinc_{ref}$ para identificação dos sinais reais de sincronismo horizontal *hsinc* presentes no sinal de vídeo composto; neste caso também existe *jitter*, porém este é variável linha a linha e pode ser reduzido a níveis bastante aceitáveis desde que se aprimore a eletrônica do circuito PLL. O método ideal de sincronização (sincronismo de pixel) utiliza o sinal de temporização do CCD como referência para a localização dos sinais de sincronismo horizontal, que são precisamente detectados por meio de um circuito capaz de identificar a variação negativa do sinal *hsinc* em um intervalo de tempo inferior a 0,5 ns (RAYNOR; SEITZ, 1990). Esta última forma de sincronização elimina toda e qualquer distorção geométrica do sinal digitalizado devida à ocorrência de *jitter* e às variações térmicas no circuito da câmera; sua implementação, porém, nem sempre é possível, pois a despeito da proliferação de hardware para aquisição de imagens baseado em sincronismo de pixel, a maioria das câmeras comerciais não transmitem o sinal de temporização do CCD.

3.2.3 Minimização dos Efeitos da Degradação do Espectro de Potência

As operações de filtragem e reconstrução realizadas pelo sistema optoeletrônico de aquisição, impõem ao espectro de potência da *imagem focalizada ideal* dois tipos de degradação: 1º) estreitamento da banda de freqüências, limitada ao menor valor da freqüência de corte dos filtros *antialiasing* do sistema; 2º) alteração de forma, com aumento da contribuição das baixas freqüências e atenuação das altas freqüências do espectro.

Informações contidas em freqüências espaciais superiores à freqüência de Nyquist são absolutamente irrecuperáveis. Quanto às alterações de forma do espectro de potência da imagem, existem alguns recursos que podem ser utilizados para minimizar os efeitos desse tipo de degradação.

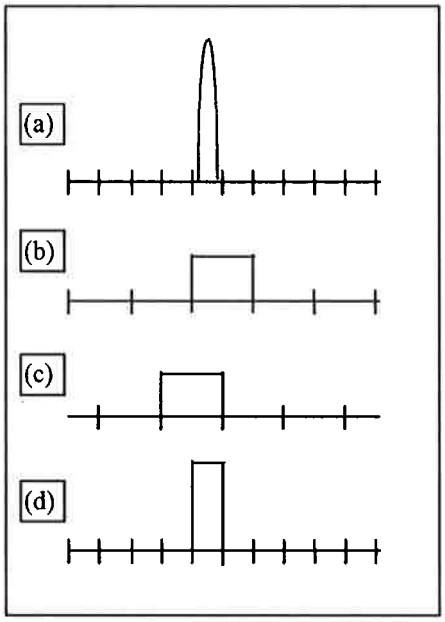

Figura 3.7 (extraída de KEREN et al, 1988): (a)Intensidade luminosa real de um "ponto" do objeto; (b) Imagem$_1$ de baixa resolução; (c) Imagem$_2$ de baixa resolução; (d) Imagem de alta resolução.

O método de super-resolução descrito em (KEREN *et al*, 1988) utiliza uma seqüência de imagens de um mesmo objeto capturadas por uma câmera móvel, para construir uma nova imagem de maior resolução. Como as diversas imagens da

seqüência são capturadas em condições ligeiramente diferentes, desde que elas sejam enquadradas com precisão de sub-pixel é possível combinar os correspondentes níveis de cinza de cada imagem para obter uma melhor estimativa dos níveis de cinza da *imagem verdadeira* (figura 3.7). Apesar de esse método não exigir que se faça nenhuma hipótese sobre o processo de degradação da imagem, a sua aplicação requer a introdução, no arranjo experimental, de dispositivos automáticos de deslocamento da platina do microscópio, que possibilitem a captura da seqüência de imagens micro transladadas / rotacionadas.

A literatura descreve inúmeros filtros não-ideais de interpolação, capazes de reconstruir uma imagem digital com mínimo erro (MARION, 1991; WOLBERG, 1990). Como, nos casos reais, a digitalização é sempre acompanhada por alguma forma de degradação do espectro de potência, a simples aplicação desses filtros às imagens digitais pode produzir imagens interpoladas de alta resolução com erros consideráveis.

SEITZ (1988) alerta para o fato de que os métodos empíricos de interpolação de imagens só produzem resultados adequados quando a forma da função interpoladora é semelhante à da função de espalhamento de ponto (*PSF*) do sistema de aquisição de imagens. Dado que, em geral, a *PSF* do sistema é desconhecida, o mesmo autor recomenda a utilização de um filtro de interpolação gaussiano, pois: 1º) na ausência de informações adicionais, a *PSF* (e também a *MTF*[3], ou seja, a Transformada de Fourier da *PSF*), sendo a resultante de operações encadeadas de filtragem, deve convergir, pelo teorema do limite central, para uma função gaussiana bidimensional; 2º) conforme originalmente apresentado em (MARR; HILDRETH, 1980), o filtro gaussiano é o que melhor resolve o conflito entre dois requisitos desejáveis: banda de freqüências estreita,

[3] *MTF*, de "Magnitude of the Transfer Function".

para prevenção de *aliasing*, e largura espacial estreita, para otimização da capacidade de localização espacial.

Recentemente, diversos trabalhos têm apresentado resultados de experimentos realizados para determinar *PSFs* de sistemas de aquisição de imagens. Utilizando como referência imagens de objetos bidimensionais apresentando borda-degrau padrão *(knife edges)*, posicionadas segundo ângulos θ variáveis entre 0 e 90° (ver figura 3.8), constroem-se curvas representativas da variação espacial de níveis de cinza ao longo da direção normal a cada borda, descrevendo-se assim o efeito da *PSF* do sistema na referida direção. A despeito das particularidades dos métodos de ajuste de curvas utilizados, os resultados experimentais publicados revelam que, em geral, as *PSFs* dos sistemas de aquisição de imagens têm forma assimétrica. ZANDHUIS *et al* (1997), por exemplo, ao identificarem a *PSF* de um sistema de captura de imagens térmicas, expuseram a sua forma visivelmente alongada segundo a direção horizontal da imagem, indicando ser esta a direção com menor conteúdo de informação. STAUNTON (1998), combinando três tipos de câmeras CCD e dois tipos de hardwares de aquisição, determinou as *PSFs* para cada uma das seis possíveis configurações do sistema. Os resultados experimentais obtidos mostraram que em apenas dois casos a *PSF* possuia forma aproximadamente simétrica; nos demais, a relação entre as freqüências de corte mínima (θ= 0°) e máxima (θ= 90°) era da ordem de 4:7.

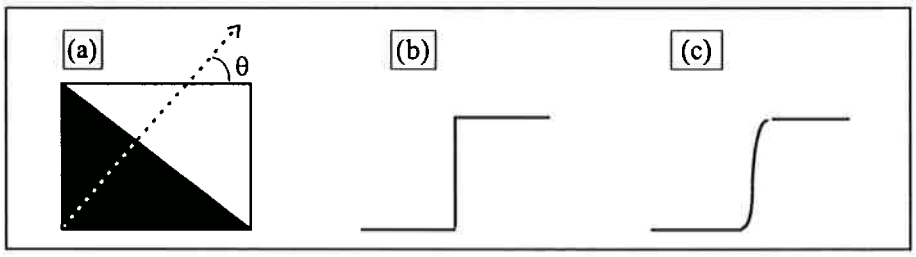

Figura 3.8: (a) Borda degrau padrão; (b) Borda ideal; (c) Borda filtrada pela PSF do sistema.

A conclusão final importante que se pode extrair desses experimentos é que a orientação do objeto observado relativamente ao sistema de referência do plano-imagem exerce substancial influência sobre a qualidade da imagem capturada e, como corolário, resulta que o filtro gaussiano perde a sua primazia perante o universo de possíveis funções interpoladoras sugeridas na literatura.

3.3 Ferramentas de Visão Computacional para Medição

Três categorias de ferramentas de medição serão aqui discutidas. A primeira delas, baseada no processo evolutivo de aquisição de conhecimento visual humano, será sempre constituída por um trio de algoritmos de medição nas escalas nominal, ordinal e racional. A segunda abordará algoritmos que emulam a habilidade humana de medir um objeto com gabaritos de medição, ou seja, padrões de referência que se ajustam (ou não) a determinadas regiões do objeto. A última categoria de ferramentas não utilizará como modelo quaisquer capacidades sensoriais ou manuais humanas — a medição se baseando tão somente nas características intrínsecas da imagem.

3.3.1 Ferramentas de Medição Baseadas no Processo Evolutivo

Conforme apresentado em (MARR, 1982), é virtualmente impossível descrever a complexidade das representações geradas pelo sistema visual humano sem que se recorra a algum arcabouço hierárquico, através do qual as informações da cena observada evoluem gradualmente desde um esboço primitivo até, quiçá, um modelo tridimensional complexo. Assim é, que, se ferramentas computacionais de medição necessitam emular habilidades visuais humanas, devem incluir algoritmos organizados de forma a gerarem uma hierarquia de medições: 1º) iniciando em escala nominal, através da classificação e agrupamento de pixels; 2º) prosseguindo em escala ordinal,

via seqüenciamento dos grupos de pixels; 3º) finalizando em escala racional, através da estimativa de parâmetros de modelos que relacionam as distâncias medidas entre os conjuntos ordenados de pixels previamente classificados.

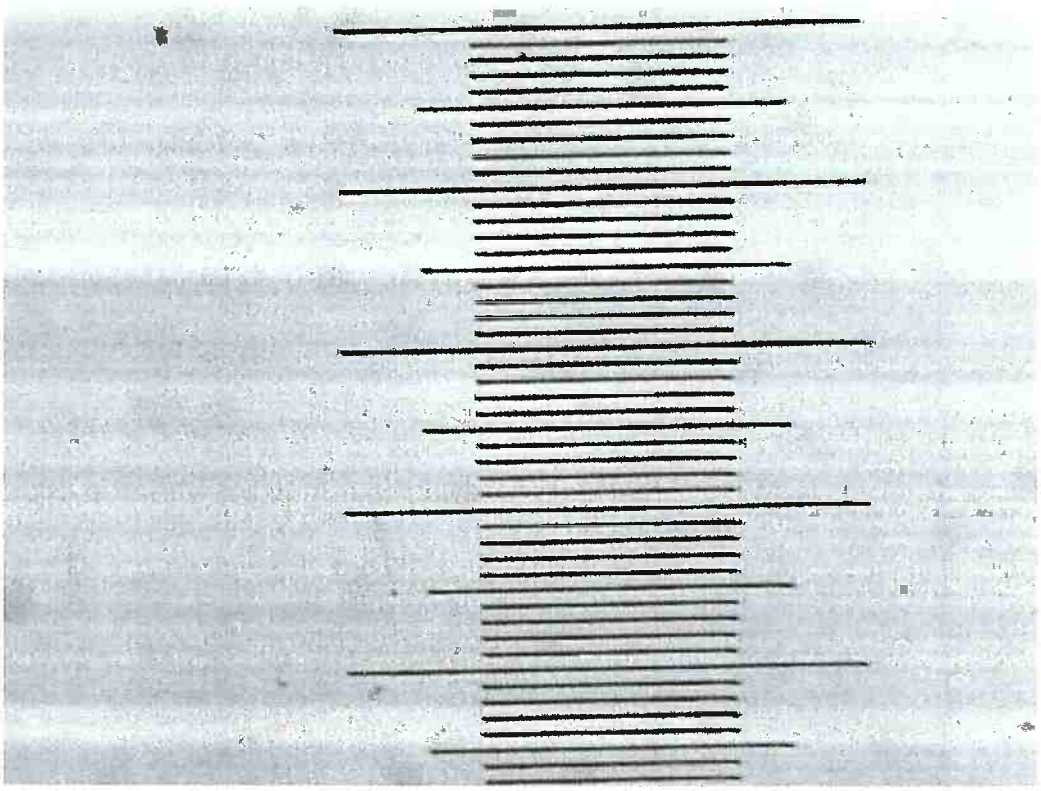

Figura 3.9: Imagem de uma microescala linear para calibração de microscópios.

No próximo ítem, serão discutidos alguns algoritmos de visão computacional capazes de emular processos de medição em diversas escalas, segundo a hierarquia acima apresentada. Em havendo necessidade de ilustração de resultados, utilizar-se-á como referência a imagem da figura 3.9, representando uma microescala para calibração de microscópios. É importante ainda frisar que, ao longo da exposição dessa hierarquia de instrumentos de medição em escalas sucessivamente cada vez mais fortes, ficará evidente a importância do ponto de vista de MARI (1997), segundo o qual medir não é apenas determinar o valor verdadeiro de uma grandeza — característica das medições em escala racional — mas às vezes atribuir um valor a uma grandeza — característica

das medições em escalas mais fracas, representando processos de tomada de decisão baseados em avaliações subjetivas.

3.3.1.1 Ferramentas de Medição em Escala Nominal

Dependendo da ampliação utilizada para produzir a imagem da microescala, duas ferramentas de medição em escala nominal — uma baseada em limiarização, a outra em detecção de bordas — podem emular o processo inicial de classificação de pixels realizado pelo sistema visual do observador. Com grande ampliação, as bordas limítrofes dos traços da escala ficam perfeitamente distinguíveis (figura 3.10-a), podendo ser utilizadas como referência para os processos subseqüentes de medição; com pequena ampliação, essas bordas se aproximam, induzindo o observador a construir mentalmente, a partir de um grupo muito maior de pixels, um *eixo medial referência* para cada traço da escala (figura 3.10-b).

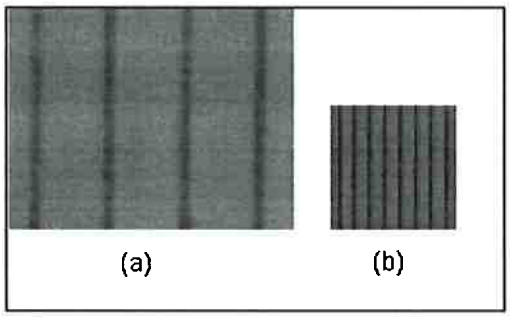

Figura 3.10 Referências de medição: (a) Bordas dos traços; (b) Eixos mediais dos traços.

Essas duas categorias de ferramentas, utilizando algoritmos de segmentação de imagens baseados em limiarização e em detecção de bordas, serão discutidas nos dois próximos tópicos a seguir.

3.3.1.1.1 Medição Nominal Baseada em Limiarização

ROSENFELD (1983) destaca o carácter estrito dos métodos de segmentação baseados em limiarização, referindo-os como ferramentas apropriadas para segmentar imagens que contêm regiões apresentando consideráveis diferenças de nível de cinza, de modo que a classificação dos pixels como pertencentes ao *background* ou a certas famílias de objetos se baseia em um vetor de características com um único elemento — o nível de cinza da imagem.

Este é exatamente o caso da imagem da figura 3.9, onde se observam traços nítidos da micro-escala contrastando com um *background* aproximadamente uniforme. Aliás, visto serem espacialmente homogêneas as distribuições de nível de cinza tanto dos objetos dessa imagem quanto do seu *background*, podem-se imediatamente descartar os vários esquemas de limiarização local e dinâmica indicados para imagens apresentando grandes variações espaciais de níveis de cinza (WESKA, 1978), concentrando-se os esforços na avaliação e seleção de métodos de limiarização global, ou seja, invariantes com a posição (x,y) de cada pixel $P(x,y)$ e independentes das propriedades locais da sua vizinhança $V\{P(x,y)\}$.

Métodos de limiarização global são processos de classificação que utilizam certas estatísticas dos níveis de cinza da imagem para a estimação dos limiares de nível de cinza que melhor agrupam os pixels, segundo um critério escolhido, em categorias de objetos conhecidas *a priori*. Em geral, interessa apenas efetuar a separação objeto-*background*, o que requer identificar, evidentemente, um único limiar.

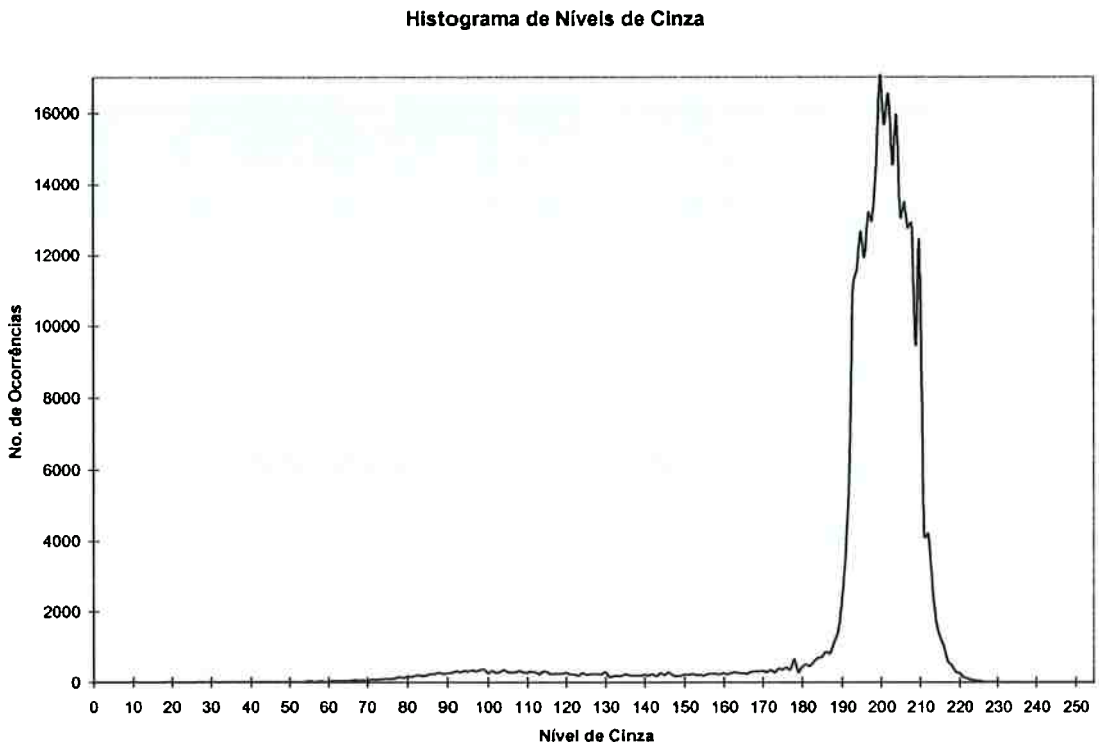

Figura 3.11: Histograma de níveis de cinza da imagem da figura 3.9.

A grande maioria dos métodos de limiarização global descritos na literatura baseia-se em estatísticas derivadas do histograma de níveis de cinza da imagem. Para o caso elementar em que este apresenta sua forma bimodal típica, revelando inexistência de interação entre os níveis de cinza do objeto e do *background*, qualquer valor de limiar selecionado no vale entre os dois lobos do histograma segmenta a imagem igualmente bem. Todavia, nos casos mais gerais em que o histograma de níveis de cinza se afasta dessa forma trivial, a seleção automática do limiar torna-se algo bem mais difícil. Analisando-se o histograma de níveis de cinza da imagem 3.9, (figura 3.11), nota-se que não há separação entre os lobos relativos às contribuições do objeto e do *background* e que estas, por sua vez, são muito desequilibradas. Em outras palavras, trata-se de um histograma unimodal onde um grande lobo concentra as contribuições do *background* e uma longa cauda as do objeto.

Dentre os inúmeros métodos de limiarização global citados na literatura, para a segmentação de imagens apresentando histogramas unimodais como esse, um, em particular, é bastante recomendado. Trata-se do método de Otsu (OTSU, 1979), que se fundamenta no seguinte princípio: "o limiar k que promove a melhor separação entre as duas classes consideradas de níveis de cinza deve ser o limiar ótimo k^*". Assim é que, aplicando-se princípios de classificação estatística de padrões, o valor ótimo do limiar, para uma imagem com n níveis de cinza, é o inteiro que maximiza um de três possíveis critérios, a saber:

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2} \qquad \eta = \frac{\sigma_B^2}{\sigma_T^2} \qquad \kappa = \frac{\sigma_T^2}{\sigma_W^2} \tag{3-2}$$

onde σ_W, σ_B e σ_T são, respectivamente, a variância interna à classe, a variância entre classes e a variância total [4]. Embora esses três critérios sejam equivalentes, o terceiro deles conduz a uma formulação bastante direta para a determinação do limiar ótimo k^*, uma vez que a variância total σ_T independe de k e a variância entre classes σ_B pode ser expressa em função apenas dos valores médios das classes. O limiar ótimo k^* é, então, expresso, por:

$$\sigma_B^2(k^*) = \max_{1 \le k \le n} \sigma_B^2(k) \tag{3-3}$$

$$\text{onde } \sigma_B^2(k) = \frac{\left[\mu_T \omega(k) - \mu(k)\right]^2}{\omega(k)\left[1 - \omega(k)\right]} \tag{3-4}$$

sendo $\omega(k)$ a área do histograma entre os níveis de cinza 0 e k;

$\mu(k)$ o momento de primeira ordem do histograma entre os níveis de cinza 0 e k;

μ_T o momento de primeira ordem do histograma entre os níveis de cinza 0 e n.

[4] Evidentemente as classes referidas são a do objeto e do *background*.

A adoção do método de limiarização de Otsu para a segmentação da imagem 3.9 não é arbitrária. SAHOO *et al* (1988) realizaram experimentos para avaliar 9 diferentes métodos de limiarização global, utilizando três imagens-teste cujos histogramas eram: 1º) bimodal típico, com dois grandes lobos; 2º) quase unimodal, com um grande lobo, uma longa cauda e um pequeno lobo sobre esta; 3º) bimodal, porém ruidosa. Os métodos foram comparados com base em pontuações geradas por duas métricas — U, medindo a uniformidade dos níveis de cinza das duas classes e F medindo o número de mudanças de forma verificadas entre as imagens original e segmentada. Os resultados dos testes deram ao método de Otsu a maior pontuação: o primeiro posto nos três casos, em relação a U e, quanto a F, leve perda de desempenho no caso do histograma ruidoso.

Apesar de o método de Otsu continuar a ser apontado como uma ferramenta de limiarização automática bastante robusta, servindo inclusive como plataforma para o desenvolvimento de métodos de limiarização mais sofisticados (GUO; PANDIT, 1998), sua formulação original baseia-se na hipótese de que σ_B^2 possui um único máximo, a qual, conforme apontado em (KITTLER; ILLINGWORTH, 1985) pode não se verificar para determinadas imagens apresentando objetos muito pequenos sobre o *background*. Nessas situações, σ_B^2 pode apresentar dois ou mais pontos de máximo local, dos quais o máximo global não necessariamente corresponde ao valor ideal de limiar.

A correta implementação do método de Otsu deve incluir um ciclo de verificação de existência de máximos locais da função σ_B^2 e, para o caso de esses existirem, algum mecanismo de seleção do máximo local mais adequado, seja através de critério heurístico automático, seja via intervenção do operador humano.

A imagem da figura 3.12 é o resultado da binarização da imagem da figura 3.9 utilizando-se o valor 165 como limiar de nível de cinza, obtido automaticamente através do método de Otsu.

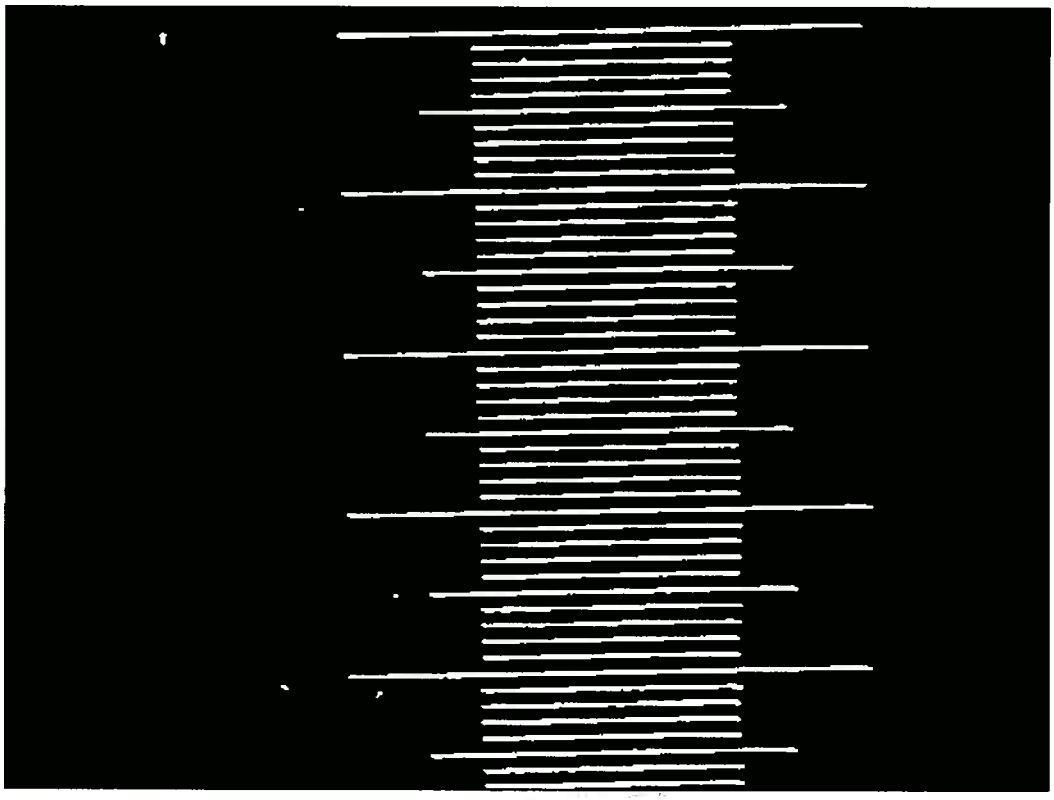

Figura 3.12: Limiarização da imagem da figura 3.9 pelo método de Otsu.

Após a aplicação do algoritmo de limiarização, os pixels da imagem original apresentam uma entre duas possíveis medidas da escala nominal de medidas {*objeto, background*}. Os *pixels-background* são completamente descartados, enquanto que os *pixels-objeto* podem ser diretamente utilizados nos processos de medição em escalas mais fortes, ou, então, submetidos a processo de medição refinada em escala nominal. Neste último caso, a escala natural de medidas nominais é definida por dois valores — {*eixo-medial, periférico*}, ou seja, o pixel pertence ao eixo medial do objeto ou é periférico a este.

Conforme salientado em (HARALICK; SHAPIRO, 1991), uma descrição não matemática do eixo medial de uma região plana pode ser apresentada utilizando-se a metáfora do campo incendiado: "ateando-se fogo ao longo do contorno de uma região coberta de vegetação, as frentes de chama do incêndio serão extintas sobre o seu eixo medial". Existem na literatura vários algoritmos que podem ser utilizados para localizar

o eixo medial de objetos de imagens binárias. Dentre estes destacam-se os algoritmos de esqueletonização baseados em operadores morfológicos, compactos e facilmente implementáveis em ambientes onde os operadores básicos *dilatação* e *erosão* já se encontram disponíveis. A expressão 3-5, por exemplo, descreve um tal algoritmo, que gera o esqueleto de uma imagem binária quando a seqüência recursiva de operações morfológicas de afinamento atinge a condição de idempotência [5].

$$A_{n+1} = \left(\cdots\left\{\left[A_n \varnothing(J_1, K_1)\right] \varnothing \cdots \varnothing(J_8, K_8)\right\}\right) \tag{3-5}$$

onde A_n é o esqueleto da imagem binária na $n^{\underline{ésima}}$ iteração.

$A \varnothing(J,K) = A - B$ é uma operação de afinamento da imagem binária, que remove os pixels B localizados pelo operador *hit-and-miss* ($A \otimes(J,K) = B$).

$A \otimes(J,K)$) é a implementação do casamento de padrões entre a imagem original A e uma assim chamada *vizinhança de Golay* (figura 3.13) representada pelo par de elementos estruturantes J (do objeto) e K (do *background*), os quais identificam de forma única determinadas regiões do objeto na imagem.

(J_i,K_i), para i=1...8, são as rotações de 0° a 315° da vizinhança de Golay original.

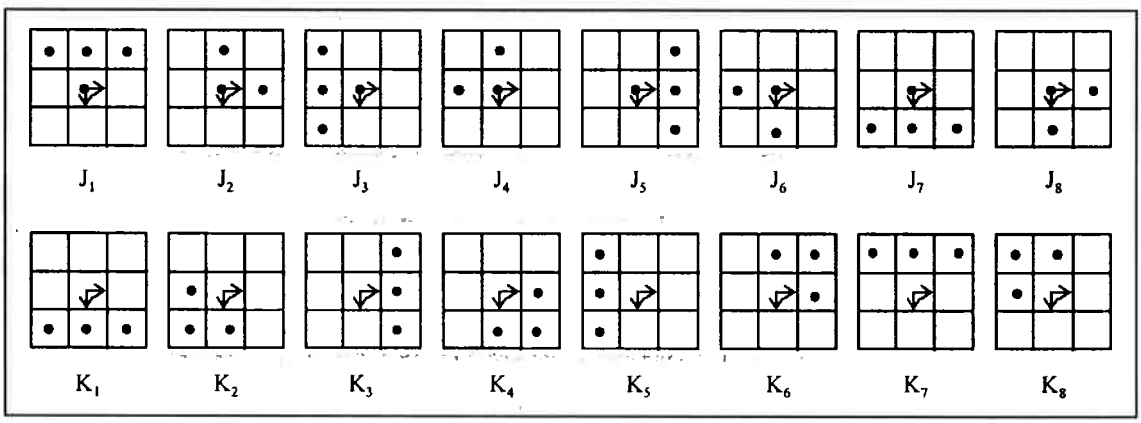

Figura 3.13: Vizinhanças de Golay do tipo L rotacionadas.

[5] Ou seja, quando a n+1$^{\underline{ésima}}$ aplicação do operador não altera os resultados obtidos na n$^{\underline{ésima}}$ aplicação.

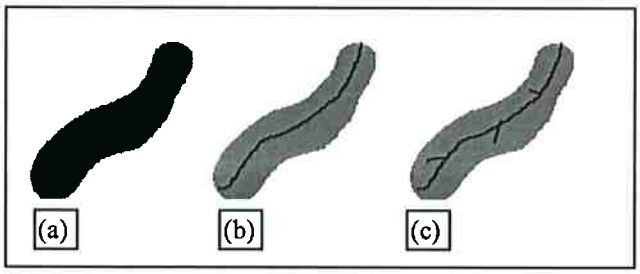

Figura 3.14: (a) Objeto binário original; (b) Esqueleto ideal; (c) Esqueleto deformado.

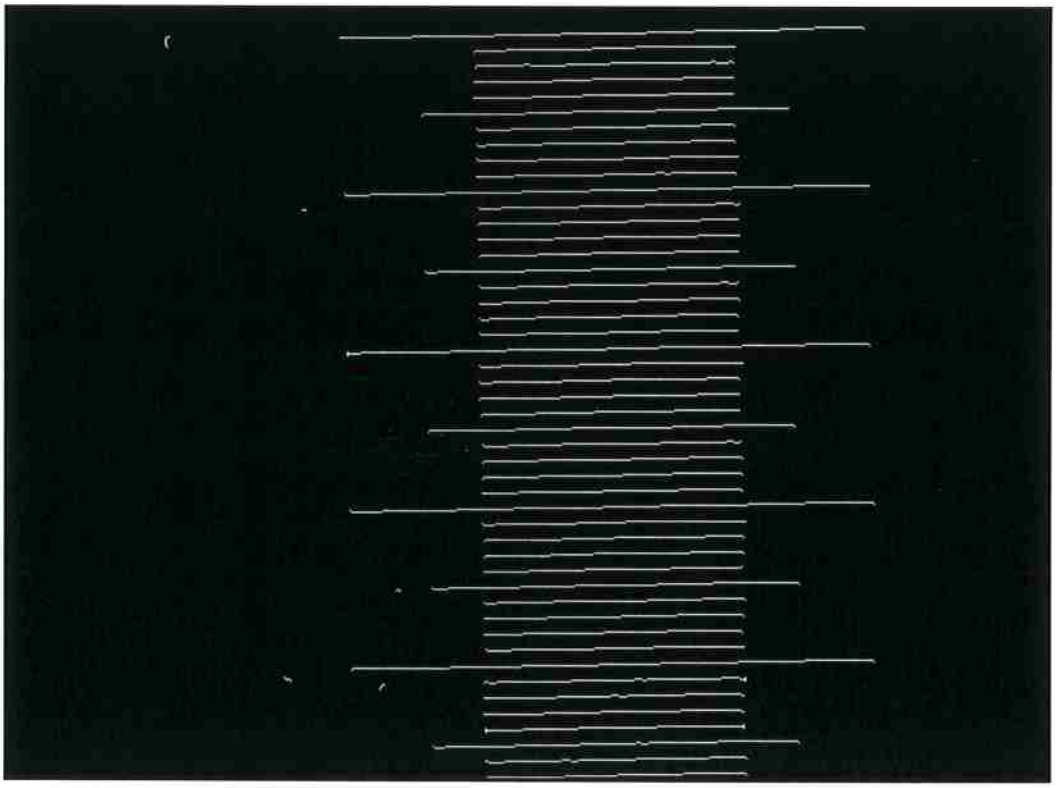

Figura 3.15: Esqueletonização da imagem da figura 3.12.

Certas precauções, porém, devem ser tomadas em relação a algoritmos como esse: conforme se observa na figura 3.13, a seqüência de operações de afinamento apresentada na expressão 3-5, utiliza 8 vizinhanças de Golay do tipo *L* (SERRA, 1989) rotacionadas de ângulos de 45°. Como, em geral, a imagem é mapeada em uma grade retangular e não em uma grade hexagonal isométrica ideal, rotações ímpares múltiplas de 45° deformam a geometria original da vizinhança de Golay, o que pode dar origem a um esqueleto com pequenos "ramos" atrelados ao eixo principal (figura 3.14). Em tais

circunstâncias, é necessário aplicar uma nova seqüência de afinamentos utilizando uma vizinhança de Golay do tipo *E*, indicada para a eliminação desses "ramos" indesejáveis.

A figura 3.15 apresenta os eixos mediais dos traços da imagem binária da figura 3.12, determinados através de um algoritmo morfológico de esqueletonização utilizando vizinhança de Golay do tipo *L*.

3.3.1.1.2 Medição Nominal Baseada em Detecção de Bordas

Em processos de segmentação e interpretação de imagens, bordas de luminância desempenham um papel crucial, uma vez que representam características diretamente relacionadas com a geometria da cena, tais como: descontinuidades na reflectância e orientação das superfícies dos objetos, sombras geradas pelas fontes de iluminação e oclusões parciais de objetos situados em diferentes profundidades. As várias centenas de algoritmos de detecção de bordas já publicados não apenas realçam a sua importância em sistemas de visão computacional como, principalmente, evidenciam o carácter mal definido (*ill posed*) do problema e a natureza não universal das soluções propostas, muitas vezes "validadas" com base em um número insignificante de imagens. Neste contexto, há dois caminhos seguros para se integrar detecção de bordas ao processo de segmentação da imagem: 1º) Utilização de um detector de bordas clássico e genérico, com desempenho comprovado relativamente aos critérios de interesse, seguindo a máxima de que "é preferível utilizar um algoritmo simples com desempenho previsível a um algoritmo supostamente superior, mas complexo e de desempenho desconhecido" (COURTNEY *et al*, 1997); 2º) Desenvolvimento de um detector de bordas especial para a categoria de imagens em análise, lembrando que a agregação de conhecimento *a priori* sobre certas características da cena aumenta o poder discriminante de algoritmos de reconhecimento de padrões.

A construção das ferramentas de medição em escala nominal baseadas em detecção de bordas seguirá ambos esses caminhos. Com relação aos algoritmos clássicos, serão considerados apenas os assim chamados detectores *locais* e *regionais*, que se baseiam, respectivamente, nas duas definições tradicionais de *borda* — "objeto vizinho a variações locais de nível de cinza" ou "fronteira entre duas regiões homogêneas com diferentes níveis de cinza" (LEVIALDI, 1983). Quanto ao detector de bordas especial, por ele necessitar de informações estruturais da cena que somente estarão disponíveis após a medição utilizando ferramentas baseadas na escala ordinal, a sua apresentação será feita em seqüência à descrição daquelas ferramentas.

Métodos locais de detecção de bordas, utilizando operações de diferenciação para identificar as variações locais de nível de cinza, são particularmente sensíveis à presença de ruído na imagem, razão pela qual tanta ênfase tem sido dada na literatura à necessidade de se aplicar uma operação de suavização antes da diferenciação da imagem (TORRE; POGGIO, 1986). Ademais, como nem todas as variações de nível de cinza detectadas pelos operadores locais correspondem a bordas significativas para a descrição da cena, muitas vezes é preciso aplicar a técnica originalmente proposta em (MARR; HILDRETH, 1980), utilizando as bordas detectadas por operadores LoG (Laplacian of Gaussian) ou similares, com escalas de resolução variáveis para, através da combinação daquelas informações, se gerar uma descrição adequada da cena.

Ora, a imagem da figura 3.9 representa uma cena extremamente simples, onde uma única escala de resolução espacial é suficiente para identificar as linhas limítrofes dos traços da microescala de calibração; é suficiente, portanto, um único detector de bordas com largura fixa. Todavia, é importante frisar que aqueles traços, por serem bastante finos (3 a 4 pixels), aproximam-se bastante bem do modelo *linha* (figura 3.16-a), enquanto que os detectores locais de bordas são normalmente baseados no modelo

degrau (figura 3.16-b). Uma forma de resolver esse dilema é aplicar um detector local de bordas com largura limitada a 3 pixels, pois em caso contrário, tanto o valor máximo da magnitude do gradiente [6] de nível de cinza quanto a sua localização seriam afetados pela presença de uma segunda borda de sinal oposto na mesma vizinhança do operador. Alternativamente, pode-se utilizar a metodologia proposta por NALWA (1987), segundo o qual bordas do tipo *linha* podem sempre ser identificadas por detectores de bordas do tipo *degrau*, desde que a imagem seja previamente reconstruída através de algum processo adequado de interpolação.

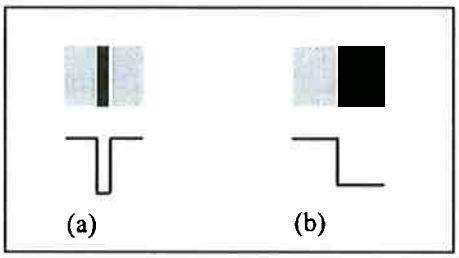

Figura 3.16: (a) Borda do tipo " linha"; (b) Borda do tipo "degrau".

Em ambas as hipóteses, seja utilizando a imagem original, seja a interpolada, fator essencial para a seleção do detector de bordas é a relação sinal/ruído da imagem. A estimativa do ruído branco σ_n em imagens [7] pode ser feita aplicando-se a formulação:

$$\sigma_n = \sqrt{\frac{1}{m} \times \sum_{i=1}^{m} \left(P_i - p_i\right)^2} \qquad (3\text{-}6)$$

onde m é o número de pixels da imagem e P_i e p_i são, respectivamente, os níveis de cinza dos pixels da imagem original e de sua versão suavizada por um filtro espacial *média* de largura escolhida.

[6] Ou de sua derivada, caso se utilizem operadores que detectam mudanças de sinal ("zero crossings") do gradiente de nível de cinza, como por exemplo, o operador LoG.
[7] Desvios nos níveis de cinza da imagem real relativamente a sua contraparte idealizada, consistindo de objeto e *background*.

A justificativa para aplicação deste método apoia-se nos resultados dos experimentos realizados por OLSEN (1993), que lhe deram máxima pontuação face a outros 5 métodos de estimação mais sofisticados. Portanto, aplicando-se à imagem da figura 3.9 a formulação 3-6, com filtro *média* de largura 3, obtém-se uma estimativa de ruído branco σ_n = 5,48. Por outro lado, calculando-se os valores médios do nível de cinza dos objetos (101,1) e do *background* (200,8) registra-se um contraste h entre ambos da ordem de 100, de onde se deduz que essa imagem apresenta elevada relação sinal/ruído: $h^2/\sigma_n^2 = 331$.

A figura de mérito proposta em (ABDOU; PRATT, 1979) avalia a qualidade de diversos detectores locais de bordas — Roberts, Sobel, Prewitt e Kirsch, entre outros — em função de sua *capacidade de detecção* e da *exatidão de posicionamento* das bordas localizadas. As curvas apresentadas nesse trabalho, baseadas em imagens de bordas do tipo *degrau* verticais e inclinadas, com níveis decrescentes de ruído, indicam que, para valores de h^2/σ_n^2 superiores a 100, praticamente inexistem diferenças de qualidade entre os detectores testados. Por outro lado, a figura de mérito utilizada em (KITCHEN; ROSENFELD, 1981), penalizando bordas grossas e descontínuas, demonstra nítida inferioridade do detector de bordas de Roberts relativamente aos demais.

Para a imagem da figura 3.9, a utilização de quaisquer dos detectores de bordas de Sobel, Prewitt ou Kirsch na construção de ferramenta de medição em escala nominal, embora se aproxime de um simples processo de tomada de decisão (portanto, uma *medição por atribuição*, segundo MARI (1997)), não é de todo arbitrária, em vista das justificativas apresentadas anteriormente.

Aplicando-se o operador de Sobel à imagem da figura 3.9 apenas realçam-se as suas bordas, conforme se observa na figura 3.17. O processo de classificação dos pixels na escala de valores nominais (*borda_esquerda, borda_direita, não-borda*) requer ainda

algumas operações e considerações adicionais. Como a estrutura dos objetos dessa imagem é extremamente simples, para identificar os pixels das bordas superior e inferior de cada traço é suficiente analisar o sinal da componente vertical do vetor gradiente a cada passo da convolução. Concluída essa separação prévia, dois processos se fazem ainda necessários: limiarização e afinamento da imagem de bordas resultante.

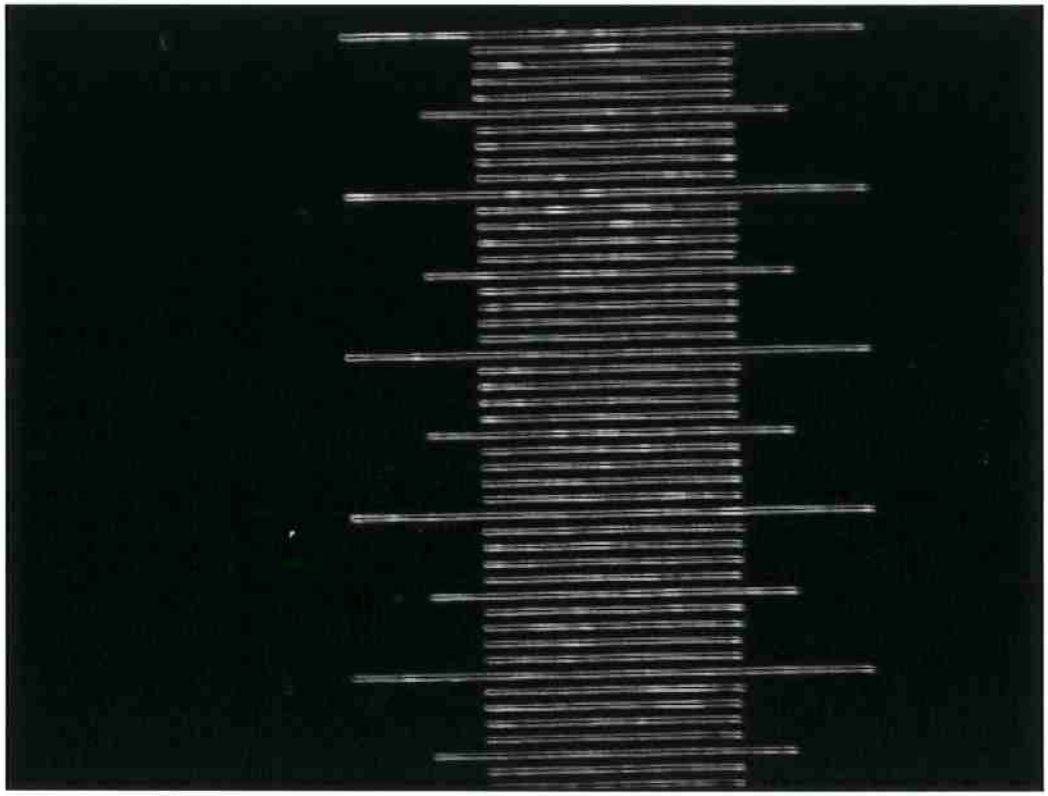

Figura 3.17: Bordas da imagem da figura 3.9 realçadas pelo operador de Sobel.

Conforme se apresenta em (ROSIN, 1997), o histograma de níveis de cinza do gradiente de uma imagem *I* contaminada com ruído branco σ, segue uma distribuição de Rayleigh, da forma:

$$P(\|\nabla I\|) = \frac{\|\nabla I\|}{\sigma^2} e^{-\frac{\|\nabla I\|^2}{2\sigma^2}} \qquad (3\text{-}7)$$

onde ∇I é o vetor gradiente de níveis de cinza da imagem *I*.

HADDON (1988), porém, já demonstrara que, quando o gradiente é estimado com auxílio de detectores locais de bordas, a distribuição 3-7 anterior pode ser aproximada por uma distribuição normal:

$$P(\|\nabla I\|) = \frac{1}{2\pi s^2} e^{-\frac{\|\nabla I\|^2}{2s^2}} \tag{3-8}$$

onde a estimativa da variância s depende apenas da variância σ do ruído branco da imagem original I e dos valores constantes das máscaras de convolução do detector de bordas. Para o caso, por exemplo, de um detector de Sobel, tem-se $s^2=12\sigma^2$. Sendo conhecida a distribuição de níveis de cinza da imagem de bordas, o valor do limiar de binarização pode ser calculado em função da proporção admissível de falsas bordas:

$$P(\|\nabla I > T\|) = \int_0^{2\pi} \int_T^{\infty} P(\|\nabla I\|) = 2e^{-\frac{T^2}{2s^2}} \tag{3-9}$$

de modo que $$T = s\sqrt{-2\ln\left(\frac{1}{2}P(\|\nabla I > T\|)\right)} \tag{3-10}$$

Essa categoria de métodos de limiarização de imagens de bordas, baseada na identificação do parâmetro σ da sua distribuição de níveis de cinza, foi a que apresentou, nos experimentos realizados por ROSIN (1997), o melhor desempenho. Todavia, não se pode deixar de destacar que, na formulação 3-10 acima, a necessidade de se especificar a fração admissível de falsas bordas introduz uma componente altamente subjetiva no processo de medição nominal.

Na figura 3.18, as bordas superiores da figura 3.17 são previamente identificadas, após o quê a imagem é binarizada utilizando-se um valor de limiar calculado pelo método exposto acima, admitindo-se uma fração de 1% de falsas bordas.

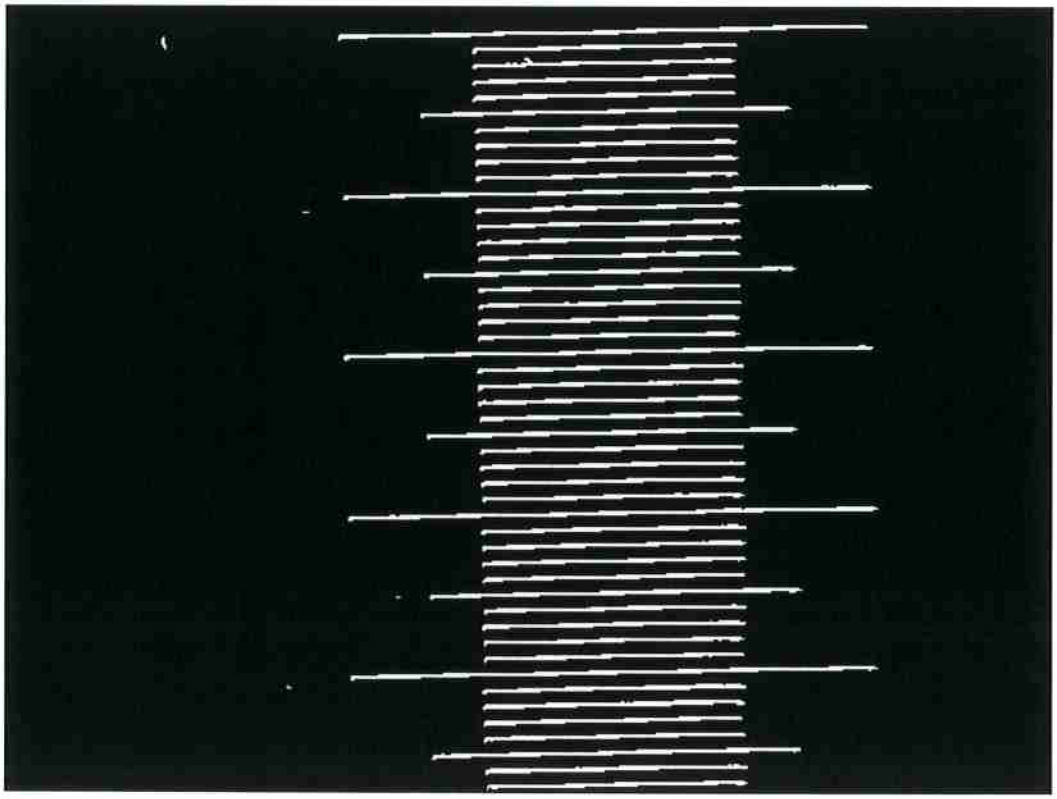

Figura 3.18: Limiarização da imagem de bordas superiores da figura 3.17.

A aplicação de detectores baseados no gradiente de níveis de cinza produz imagens com bordas espessas, conforme se observa na figura 3.18. Por esse motivo, é fundamental que a medição em escala nominal seja refinada mediante a aplicação de algum método de afinamento das bordas. De acordo com (NEVATIA; BABU, 1980), a forma mais simples e efetiva de se obter bordas finas consiste em identificar os máximos locais da imagem de bordas original (não limiarizada), o que pode ser realizado através da seguinte técnica heurística: "Um pixel P da imagem de bordas é um máximo local se: 1º) os módulos do gradiente de níveis de cinza de seus dois pixels vizinhos Q e R, na direção normal à borda, forem inferiores à de P; 2º) as direções do gradiente de níveis de cinza em Q, R forem aproximadamente iguais às de P".

Na figura 3.19 observa-se o resultado final do processo de medição em escala nominal envolvendo: a) realce das bordas através de detector de Sobel; b) separação das bordas superior e inferior (conforme indicam os dois níveis de cinza utilizados para

representá-las), mediante análise do sinal dos componentes *y* dos vetores gradiente; c) limiarização utilizando valor de limiar calculado pelo método de HADDON (1988); d) afinamento segundo a técnica heurística de NEVATIA *et* BABU (1980).

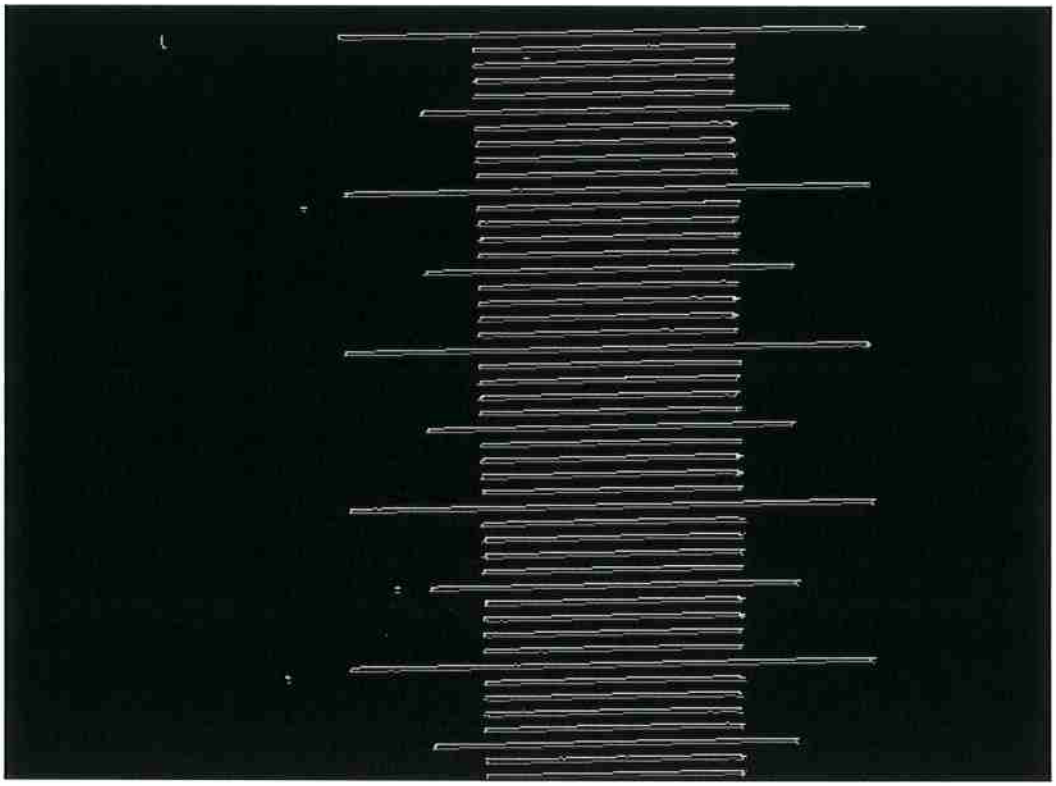

Figura 3.19: Bordas superiores e inferiores identificadas, afinadas e limiarizadas.

O processo de medição anterior poderia, em princípio, ser aprimorado, se na etapa *a*, em lugar de um detector de bordas *local,* se utilizasse um detector *regional.* Esta classe de operadores, que considera as bordas como fronteiras retas entre duas regiões com níveis homogêneos de cinza, tem como principal atrativo a habilidade de localização com precisão de subpixel. Da mesma forma que a *Teoria do Sinal Médio Local*, antes referida, estipula que a hiperacuidade visual humana seria o resultado da combinação de informações locais coletadas pelos fotossensores da retina, assim também procedem os detectores regionais ao admitirem que, ao longo de uma "região de influência", geralmente circular e de raio arbitrário, coexistem ambas as informações

da borda digitalizada real $B(x,y)$ e de um seu modelo contínuo, cujos parâmetros são identificáveis a partir de critérios estabelecidos *a priori*. Em HUECKEL (1973), por exemplo, esses parâmetros são estimados resolvendo-se um problema de minimização da diferença quadrática entre as funções digital $B(x,y)$ e o modelo $\mathcal{B}(x,y,\rho,\theta,c_1,c_2)$ contínuo (figura 3.20), sobre o círculo $\mathcal{C}$ de raio $r=4$ pixels:

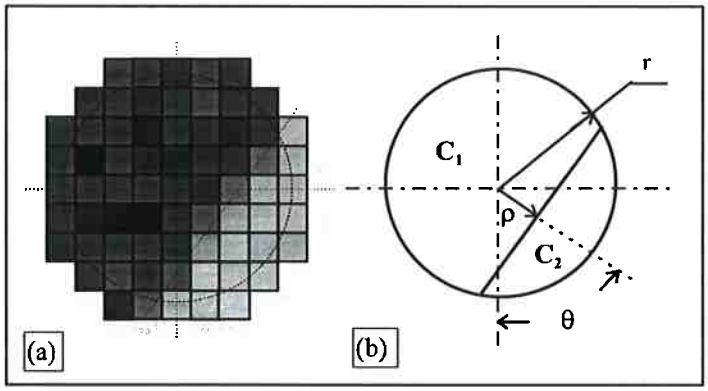

Figura 3.20: (a) Borda digitalizada real; (b) Modelo contínuo idealizado.

$$\Lambda(x,y,\rho,\theta,c_1,c_2) = \iint_{\mathcal{C}} \left[B(x,y) - \mathcal{B}(x,y,\rho,\theta,c_1,c_2) \right]^2 dx \cdot dy \qquad (3\text{-}11)$$

onde ρ,θ são as coordenadas polares da borda e c_1,c_2 os níveis de cinza das duas regiões fronteiriças.

O fato de o detector de bordas proposto por Hueckel, entre outras razões apontadas, requerer janelas de convolução bastante largas, tem estimulado o projeto de novos detectores regionais baseados em janelas de menor tamanho. JENSEN *et* ANASTASSIOU (1995) seguiram de perto as idéias de HUECKEL, desenvolvendo um detector de bordas 3x3 com precisão de subpixel; o trabalho desses autores, porém, não é acompanhado de um estudo de desempenho do detector relativamente à exatidão de localização das bordas. LYVERS *et al* (1989), por seu turno, projetaram um detector regional 5x5, que identifica os parâmetros ρ, θ, c_1, c_2 igualando os momentos de ordem 0 a 2 da distribuição de níveis de cinza da imagem digitalizada B e de seu modelo $\mathcal{B}$:

$$M_{pq} = \iint_{\mathcal{C}} x^p y^q f(x,y) dx \cdot dy \tag{3-12}$$

onde os momentos M_{pq} do modelo idealizado são determinados através da convolução da imagem original com as máscaras

$$M_{00} = \begin{matrix} .0219 & .1231 & .1573 & .1231 & .0219 \\ .1231 & .1600 & .1600 & .1600 & .1231 \\ .1573 & .1600 & .1600 & .1600 & .1573 \\ .1231 & .1600 & .1600 & .1600 & .1231 \\ .0219 & .1231 & .1573 & .1231 & .1219 \end{matrix} \qquad M_{11} = \begin{matrix} -.0098 & -.0352 & .0000 & .0352 & .0098 \\ -.0352 & -.0256 & .0000 & .0256 & .0352 \\ .0000 & .0000 & .0000 & .0000 & .0000 \\ .0352 & .0256 & .0000 & -.0256 & -.0352 \\ .0098 & .0352 & .0000 & -.0352 & -.0098 \end{matrix}$$

$$M_{10} = \begin{matrix} -.0147 & -.0469 & .0000 & .0469 & .0147 \\ -.0933 & -.0640 & .0000 & .0640 & .0933 \\ -.1253 & -.0640 & .0000 & .0640 & .1253 \\ -.0933 & -.0640 & .0000 & .0640 & .0933 \\ -.0147 & -.0469 & .0000 & .0469 & .0147 \end{matrix} \qquad M_{01} = \begin{matrix} .0147 & .0933 & .1253 & .0933 & .0147 \\ .0469 & .0640 & .0640 & .0640 & .0469 \\ .0000 & .0000 & .0000 & .0000 & .0000 \\ -.0469 & -.0640 & -.0640 & -.0640 & -.0469 \\ -.0147 & -.0933 & -.1253 & -.0933 & -.0147 \end{matrix}$$

$$M_{20} = \begin{matrix} .0099 & .0194 & .0021 & .0194 & .0099 \\ .0719 & .0277 & .0021 & .0277 & .0719 \\ .1019 & .0277 & .0021 & .0277 & .1019 \\ .0719 & .0277 & .0021 & .0277 & .0719 \\ .0099 & .0194 & .0021 & .0194 & .0099 \end{matrix} \qquad M_{02} = \begin{matrix} .0099 & .0719 & .1019 & .0719 & .0099 \\ .0194 & .0277 & .0277 & .0277 & .0194 \\ .0021 & .0021 & .0021 & .0021 & .0021 \\ .0194 & .0277 & .0277 & .0277 & .0194 \\ .0099 & .0719 & .1019 & .0719 & .0099 \end{matrix}$$

(3-13)

Os referidos autores demonstraram o superior desempenho de seu detector face a três outros detectores regionais, utilizando como referência o viés de deslocamento devido a efeitos de amostragem, ou seja, o deslocamento calculado das posições das bordas de um modelo contínuo de *borda degrau* e de sua correspondente versão digital, ambas localizadas pelos detectores regionais avaliados. Conforme relatado em seu trabalho, na ausência de ruído, o detector proposto apresenta erro médio quadrático de localização nulo quando a borda se situa a distâncias inferiores a 1 pixel do centro; fora dessa região, esse erro pode atingir até 0,10 pixels, na ausência de ruído, e não mais que 0,13 pixels quando a relação sinal/ruído, definida pela fórmula $20 \log_{10} h/\sigma$, alcança valores de até 50 dB. Todavia, realizando-se a convolução com o detector a intervalos de 1x1 e não 5x5 pixels, como originalmente proposto, as bordas detectadas terão sempre mínimo viés de deslocamento.

Para a imagem da figura 3-9, com relação sinal/ruído igual a $20\log_{10} 100/5{,}48 =$ 25dB, o viés teórico de deslocamento introduzido por esse operador na posição das bordas é de apenas 0,06 pixel. Contudo, é importante salientar que, por causa das inevitáveis diferenças entre o modelo regional adotado e as reais formas das bordas da imagem, o verdadeiro viés de deslocamento poderá ser significativamente maior. Na figura 3.21 apresenta-se o gráfico das coordenadas de uma das bordas da imagem da figura 3.9 (borda superior do primeiro traço) detectada pelo operador regional proposto por LYVERS *et al.* Os ressaltos observados ao longo do gráfico são uma manifestação de *aliasing* resultante da aplicação de um algoritmo de afinamento após as operações de realce e limiarização de bordas [8].

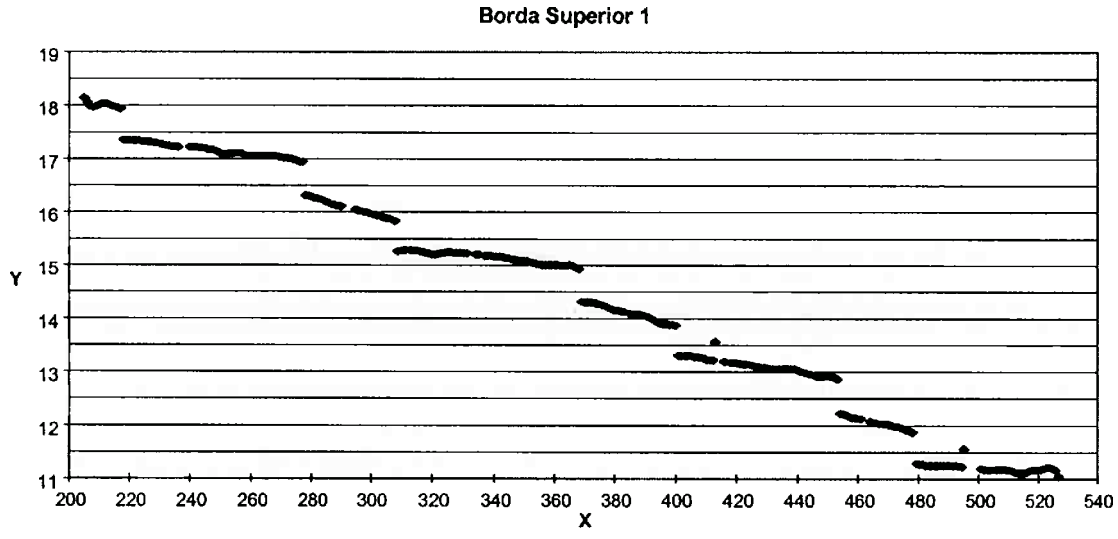

Figura 3.21: Coordenadas de bordas localizadas pelo operador de Lyvers et al.

3.3.1.2 Ferramentas de Medição em Escala Ordinal

Para que medições quantitativas possam ser realizadas, os pixels classificados por quaisquer dos processos de medição em escala nominal apresentados anteriormente

[8] Esse algoritmo, ao preservar apenas os máximos locais da imagem de bordas, alarga significativamente a banda de freqüências do seu espectro.

devem ser agrupados em torno dos objetos componentes da estrutura da cena, segundo uma ordem apropriada. No caso específico da imagem da figura 3.9, essa estrutura consiste de traços paralelos τ_i, de tipos T_1, T_2, T_3, com comprimentos decrescentes L_1, L_2, L_3 organizados segundo a seqüência $\mathcal{S}$ apresentada na figura 3.22, iniciando com um traço $\tau_1=T_1$, necessariamente, e finalizando com um traço $\tau_n=T_j$, onde j=1,2 ou 3 dependendo da distância de τ_1 à origem e de sua orientação em relação ao sistema de coordenadas da imagem.

A simplicidade, tanto da forma dos objetos, como da sua estrutura, permite enunciar de forma objetiva o problema de medição em escala ordinal: "agregar os pixels pré-classificados da imagem segmentada I_{Seg} em torno do maior número n de retas que apresentam entre si a relação ordinal definida pela seqüência $\mathcal{S}$".

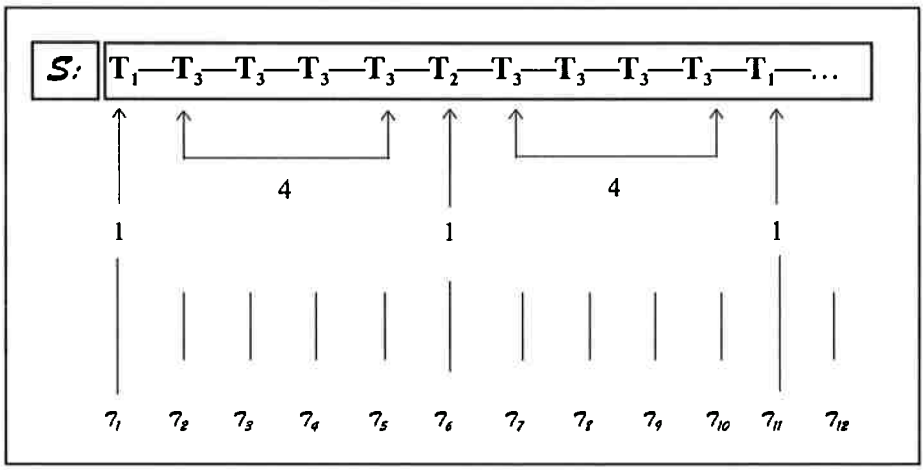

Figura 3.22: Estrutura da imagem da microescala de calibração.

Valendo-se, ora da particular forma dos objetos, ora da especial sintaxe da estrutura da cena, duas ferramentas específicas de medição em escala ordinal podem ser construídas a partir de algoritmos clássicos de visão computacional: uma, utilizando a transformada de Hough, irá identificar grupos de pixels agregados em torno de retas; a outra, baseada em algoritmo de rotulação, irá localizar seqüências desconexas de pixels conectados.

3.3.1.2.1 Medição em Escala Ordinal Usando a Transformada de Hough

Em sua forma paramétrica mais utilizada, qual seja,

$$\rho = x_m \cos\theta + y_m \operatorname{sen}\theta \tag{3-14}$$

a transformada de Hough ($\mathcal{T}$-$\mathcal{H}$) de uma imagem binária B é o espaço discreto $\mathcal{H}(\rho \times \theta)$, onde o alinhamento de n pixels $P_m(x_m, y_m)$ em B corresponde a um ponto de acumulação (máximo local) de $\mathcal{H}$ representando a intersecção de n senóides da forma 3-14, ou seja:

$$\mathcal{H}(\rho_i, \theta_i) = \sum_{x_m \cos\theta_i + y_m \operatorname{sen}\theta_i = \rho_i} 1 \tag{3-15}$$

Todavia, a quantização espacial da imagem digital e os inevitáveis ruídos associados às posições (x,y) dos pixels, fazem com que os picos do espaço $\mathcal{H}(\rho \times \theta)$ sejam afetados por espalhamento. Sendo o espaço $\mathcal{H}(\rho \times \theta)$ discreto, a seleção dos incrementos $\Delta\rho$ e $\Delta\theta$ não pode ser feita de forma arbitrária, pois: a sub-amostragem, violando o Teorema de Nyquist, acarreta fusão de picos e, portanto, perda da capacidade de discriminação de retas distintas; a super-amostragem, por outro lado, transforma cada pico característico em um grupo de pequenos máximos locais, diminuindo a capacidade de localização do algoritmo de busca.

A prevenção de *aliasing* na Transformada de Hough pode ser realizada através da filtragem da função acumulação $\mathcal{H}(\rho_i, \theta_i)$, (expressão 3-15), estendendo-se o núcleo unitário de contagem $\{(\rho_i, \theta_i)\}$ a uma "zona de votação" $\{(\rho_k, \theta_k) \mid i\text{-}w \leq k \leq i+w\}$, onde a contribuição dos pixels incidentes em $\{(\rho_i, \theta_i)\}$ recebe peso máximo, enquanto as contribuições incidentes em pontos cada vez mais distantes do centro recebem pesos sucessivamente decrescentes. PALMER *et al* (1997) propõem, em substituição ao núcleo unitário de contagem (w = 1/2)

$$K = \begin{cases} 1 & se\ |\xi| \le w \\ 0 & em\ caso\ contrario \end{cases} \tag{3-16}$$

a seguinte zona de votação robusta ($w \geq 1$):

$$K_R = \begin{cases} 1 - 2\dfrac{\xi^2}{w^4} + \dfrac{\xi^4}{w^4} & para\ |\xi| \le w \\ 0 & em\ caso\ contrario \end{cases} \tag{3-17}$$

onde, em ambos os casos, ξ_m é a distância entre um pixel $P(x_m,y_m)$ da imagem e a reta (ρ_i,θ_i) considerada, ou seja:

$$\xi_m = x_m \cos\theta_i + y_m \operatorname{sen}\theta_i - \rho_i \tag{3-18}$$

Para o caso de se conhecer *a priori* certas características das retas presentes na imagem —comprimento mínimo L e espessura máxima $2b$ — NIBLACK *et* PETKOVIC (1990) propõem um procedimento para determinação dos incrementos $\Delta\rho$ e $\Delta\theta$, de modo a estabelecer um compromisso entre evitar os problemas da sub amostragem e os da super amostragem, acima apresentados. Para tanto, fixam o valor de $\Delta\theta$ e calculam $\Delta\rho$, de forma tal que o espaço discreto $\mathcal{H}(\rho\times\theta)$ contenha um pico espalhado ao longo de s_ρ unidades de ρ, sendo s_ρ o número que, arbitrariamente estipulado, determina o equilíbrio entre aqueles dois objetivos conflitantes. Assim, resulta que, conhecidos L, b, $\Delta\theta$ e s_ρ, o incremento $\Delta\rho$ é calculado através da expressão

$$s_\rho = \left\lceil \frac{L\operatorname{sen}(\Delta\theta/2) + 2b\cos(\Delta\theta/2)}{\Delta\rho} \right\rceil + 2 \tag{3-19}$$

onde $\lceil z \rceil$ representa o maior inteiro estritamente menor que z.

As imagens segmentadas da microescala, em quaisquer das formas mostradas nas figuras 3.12 (limiarizada), 3.15 (limiarizada/esqueletonizada) ou 3.19 (bordas afinadas), têm estrutura e geometria bastante simples, o que facilita a determinação dos valores $\Delta\theta$ e $\Delta\rho$. O menor segmento de reta tem comprimento L da ordem de 150 pixels

e a maior espessura observada — caso da imagem 3.12 — é da ordem de 4 pixels. Além disso, como existe pleno controle da orientação da microescala em relação ao sistema de referência da imagem, a varredura angular, ou seja, $[\theta_{min}, \theta_{max}]$, pode ser limitada a um intervalo muito menor que [0°, 180°], diminuindo-se, assim, a extensão do espaço $\mathcal{H}(\rho \times \theta)$.

Na figura 3.23 apresenta-se a transformada de Hough da imagem das bordas superiores da figura 3.19. O intervalo de varredura de θ foi limitado a [87°,92°], de modo que ρ assume valores no intervalo:

$$[\rho_{min}, \rho_{max}] = \left[0, \quad max\langle Y_{max} \operatorname{sen}\theta + X_{max} \cos\theta \rangle_{\theta \in [\theta_{min}, \theta_{max}]}\right] \tag{3-20}$$

onde X_{max} e Y_{max} são, respectivamente, o número de colunas e linhas da imagem. O incremento $\Delta\theta$ foi arbitrado em 0,25° e $\Delta\rho$ calculado a partir da expressão 3-19, adotando-se L=150, b=2 e admitindo-se que o máximo espalhamento do pico é s_ρ= 5 unidades de ρ. O processo de acumulação de $\mathcal{H}(\rho \times \theta)$ foi realizado utilizando-se a zona de votação robusta (expressão 3-17) proposta por PALMER *et al* (1997).

Figura 3.23: Transformada de Hough das bordas superiores da imagem da figura 3.19.

No caso de imagens apresentando segmentos de reta com grandes diferenças de comprimento, a localização de picos no espaço $\mathcal{H}(\rho \times \theta)$ é uma tarefa difícil, requerendo, por exemplo, a aplicação de um método de busca adaptativa [9](RISSE, 1989). Esse, porém, não é o caso das imagens aqui consideradas, cuja relação entre os comprimentos mínimo e máximo dos segmentos é da ordem de 1:2. Para tais imagens, portanto, o

[9] Através desse método, a cada máximo local (ρ_i, θ_i) identificado, reconstrói-se $\mathcal{H}(\rho \times \theta)$ eliminando-se as contribuições dos pixels da imagem à formação da superfície em torno de (ρ_i, θ_i).

método usual de limiarização de $\mathcal{H}(\rho \times \theta)$ com um valor fixo de limiar, é suficiente para excluir os picos irrelevantes e permitir que um simples algoritmo de busca identifique os picos "verdadeiros", que denotam a presença de uma reta. Por outro lado, a esperada presença de variações residuais em torno desses picos, pode ser facilmente eliminada através do processo de interpolação proposto em (NIBLACK; PETKOVIC, 1990) que, ademais, apresenta a vantagem de permitir a determinacão das posições (ρ_i, θ_i) das retas com resolução superior à da grade $\Delta\theta\Delta\rho$.

Na figura 3.24 apresenta-se a imagem de retas correspondentes aos máximos locais (ρ_i, θ_i) identificados na imagem limiarizada de $\mathcal{H}(\rho \times \theta)$, adotando-se limiar fixo de 30% do valor do máximo global e aplicando-se o processo de interpolação acima mencionado.

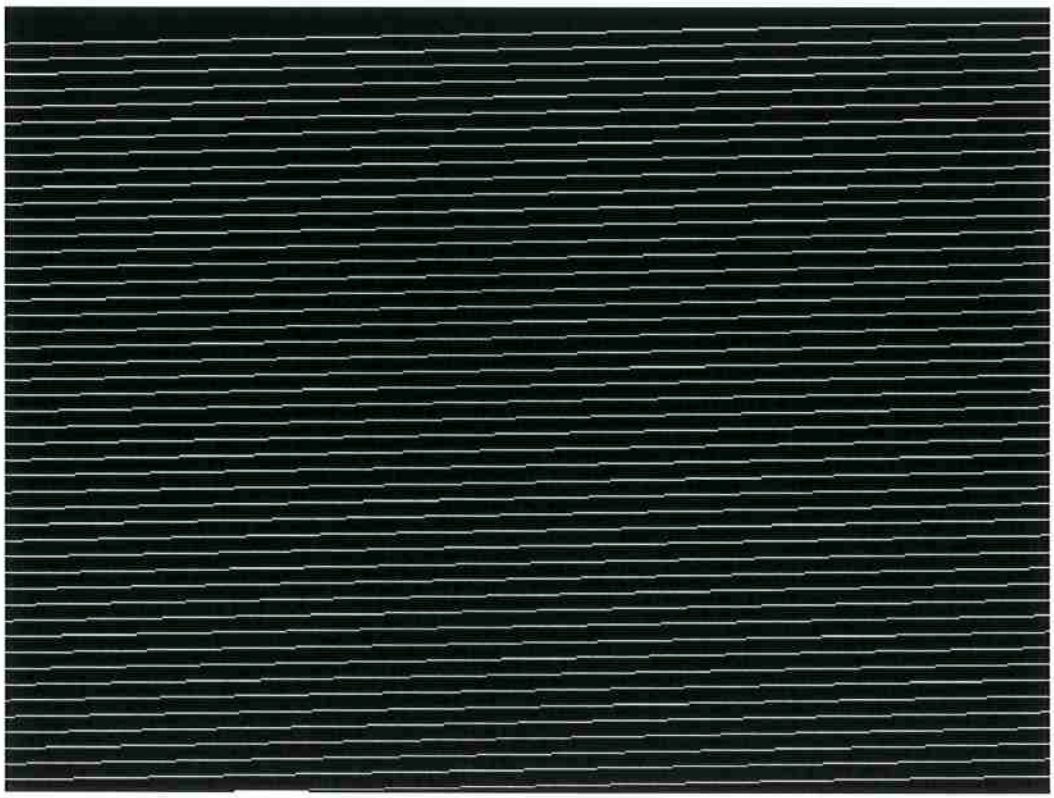

Figura 3.24: Retas correspondentes aos máximos locais da imagem da figura 3.23.

Considerando que o algoritmo de busca de máximos locais no espaço $\mathcal{H}(\rho \times \theta)$ produz uma seqüência $\mathcal{S}_{\mathcal{H}}$ de pares (ρ_i, θ_i) ordenada segundo valores decrescentes de $\mathcal{H}(\rho_i, \theta_i)$, e que, para as medições em escala racional, as retas devem estar ordenadas segundo a seqüência $\mathcal{S}$, característica da estrutura de uma imagem de microescala, é necessário aplicar um algoritmo de ordenação para reordenar $\mathcal{S}_{\mathcal{H}}$ segundo valores crescentes de ρ_i. Finalmente, conforme se ilustra na figura 3.25, de posse dos pares (ρ_i, θ_i) ordenados segundo valores crescentes de ρ_i, localizam-se na imagem segmentada I_{Seg} os diversos conjuntos ordenados de pixels segundo a seqüência $\mathcal{S}$ desejada. Para tanto, é suficiente estabelecer uma janela de busca de comprimento infinito e semi-largura r (por exemplo, r = 1,5 pixel) em torno de cada uma das retas hipotéticas (ρ_i, θ_i), e construir os conjuntos $\mathcal{T}_i$ contendo as coordenadas (x_m, y_m) dos pixels da imagem segmentada interiores a essa janela, ou seja:

$$\mathcal{T}_i = \left\{ (x_m, y_m) \xrightarrow{corresponde} P(x_m, y_m) \in I_{seg} \; e \; \left| x_m \cos\theta_i + y_m \operatorname{sen}\theta_i - \rho_i \right| \le r \right\} \quad (3\text{-}21)$$

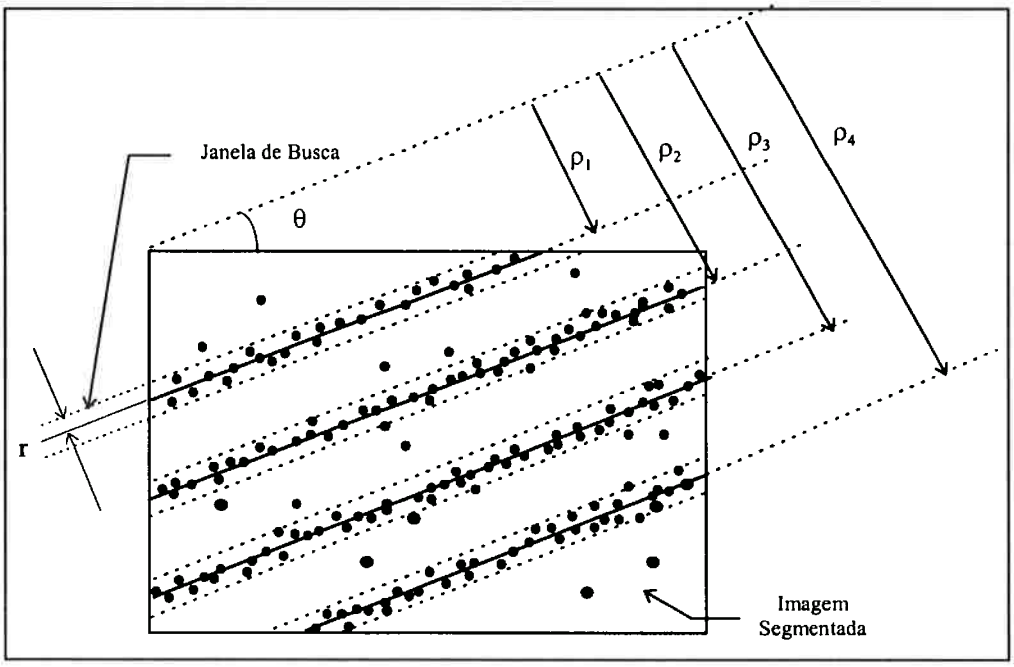

Figura 3.25: Construção dos conjuntos de pixels descritores dos traços da microescala.

O resultado final do processo de medição em escala ordinal é, portanto, a série ordenada de conjuntos $\mathcal{T}_i$ correspondendo homomorficamente à seqüência $\mathcal{S}$ de traços da microescala.

3.3.1.2.2 Medição em Escala Ordinal Usando Rotulação

Por causa da particular estrutura da imagem da microescala, é possível orientar os seus traços segundo um ângulo θ tal que a simples aplicação de um algoritmo seqüencial de rotulação de pixels conectados (HORN, 1986) produza a seqüência ordenada $\mathcal{S}$ de traços, anteriormente referida.

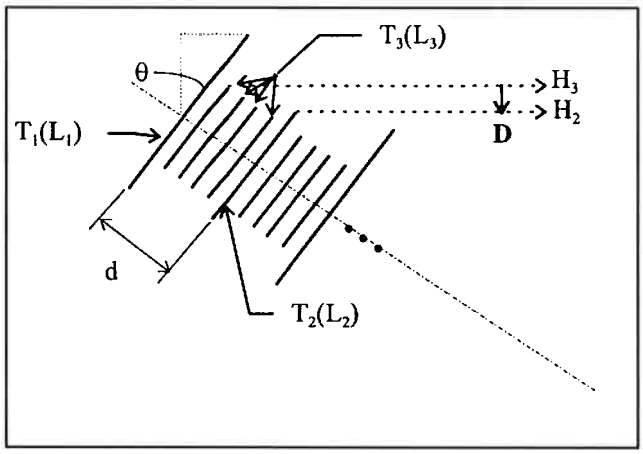

Figura 3.26: Orientação da microescala para a correta rotulação dos traços.

Varrendo-se a imagem ao longo das linhas, essa condição é satisfeita sempre que for positivo o sinal do vetor **D** (figura 3.26), entre as linhas H_2 e H_3 das extremidades dos traços T_2 e T_3 respectivamente. Em outras palavras, não devem ser localizados pixels de um novo traço T_2 (de comprimento L_2) à direita, antes que todos os demais 4 traços T_3 (de comprimento $L_3<L_2$) tenham sido previamente rotulados. Ademais, conforme se pode inferir da figura 3.26, existe um ângulo limite θ_{lim} para o qual H_2 e H_3 são coincidentes; para valores de θ inferiores a θ_{lim} basta alternar a direção de varredura da

imagem para que o algoritmo de rotulação produza a mesma seqüência $\mathcal{S}$. Da figura 3.26, extrai-se o valor de θ_{lim} a partir de

$$\frac{L_1 - L_2}{2}\cos\theta_{\text{lim}} + d\,\text{sen}\,\theta_{\text{lim}} = \frac{L_1 - L_3}{2}\cos\theta_{\text{lim}} + \frac{d}{5}\text{sen}\,\theta_{\text{lim}}$$

ou seja:

$$\theta_{\text{lim}} = \tan^{-1}\left(0.625\frac{L_2 - L_3}{d}\right) \tag{3-22}$$

de modo que, para $\theta < |\theta_{\text{lim}}|$ a varredura "por colunas" produz a seqüência $\mathcal{S}$, o mesmo ocorrendo para $\theta > |\theta_{\text{lim}}|$ quando a varredura é feita "por linhas".

Duas outras condições ainda se fazem necessárias para que a aplicação de um algoritmo de rotulação identifique e ordene os traços da imagem segundo a seqüência $\mathcal{S}$: 1ª) todos os pixels da imagem segmentada, correspondentes a um determinado traço $\mathcal{T}_i$ da microescala, devem estar conectados; 2ª) todos os pixels da imagem segmentada que não pertencem a quaisquer dos traços $\mathcal{T}_i$ da microescala devem ser descartados.

A primeira dessas condições é facilmente satisfeita aplicando-se à imagem segmentada I_{Seg} uma operação morfológica de *dilatação* com um elemento estruturante *círculo* de diâmetro maior ou igual à maior distância entre partes isoladas de um mesmo traço. Para satisfazer à segunda condição, é suficiente aplicar uma regra heurística eliminando-se todos os objetos rotulados que tenham, por exemplo, área inferior a uma certa porcentagem da área do maior objeto identificado.

Para o caso da imagem da figura 3.19, as distâncias L_2, L_3 e d são da ordem de 217, 155 e 45 pixels, respectivamente. Como as bordas estão orientadas segundo ângulo θ com a vertical próximo a 90°, e o ângulo-limite é

$$\theta_{\text{lim}} = \tan^{-1}\left(0.625\frac{217 - 155}{45}\right) \cong 41°$$

o algoritmo de rotulação deve varrer a imagem segundo a direção das linhas. A dilatação prévia da imagem de bordas superiores com um elemento estruturante *Círculo*(r=1) é suficiente para assegurar que todos os pixels de cada traço sejam conectados. Após a rotulação, selecionam-se apenas os objetos apresentando área superior a 20% da área do maior objeto encontrado. Com isso, resulta a imagem rotulada da figura 3.27, onde a progressão de níveis de cinza indica o reconhecimento individual de cada um dos traços e da sua organização seqüencial.

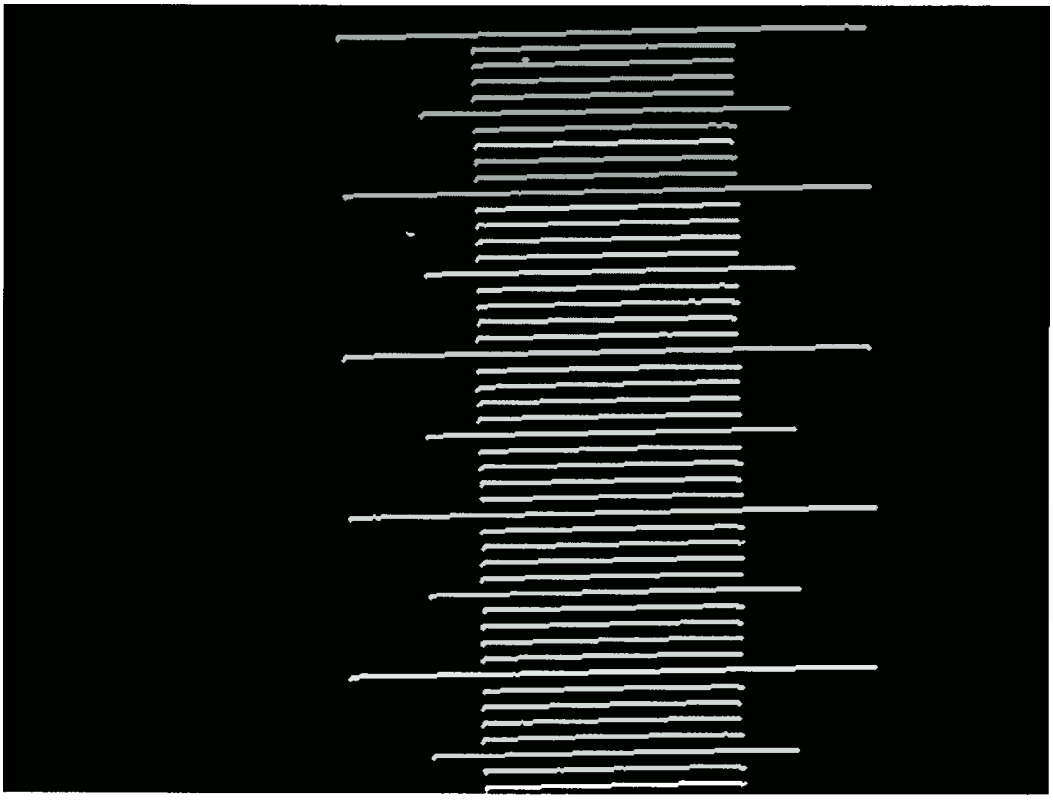

Figura 3.27: Imagem rotulada com as regiões em torno dos traços da microescala.

De posse da imagem rotulada, o processo de medição em escala ordinal é finalizado através da construção da série ordenada de conjuntos $\mathcal{T}_i$, definidos como:

$$\mathcal{T}_i = \left\{ (x_m, y_m) \xrightarrow{corresponde} coordenada\ de\ pixel\ P(x_m, y_m) \in I_{Seg} \cap Obj_i \right\} \qquad (3\text{-}23)$$

onde I_{Seg} e Obj_i são, respectivamente, a imagem segmentada e o $i^{ésimo}$ objeto da imagem rotulada da figura 3.27.

3.3.1.3 Refinamento da Medição em Escala Nominal

A medição em escala ordinal organiza os pixels em torno de uma seqüência ordenada de traços da imagem, de modo que, tomando-se como referência os conjuntos $\mathcal{T}_i$ acima descritos, pode-se avaliar a forma real de variação dos níveis de cinza da imagem na direção normal a cada traço e, assim, construir um detector especialmente ajustado para localizar bordas nessa classe de imagens. Aliás, procedimento semelhante já fora adotado por STRICKLAND *et al* (1990), que desenvolveram um método para detecção de bordas em imagens de placas de circuito impresso, consistindo de: 1º) treinamento supervisionado, mediante a seleção de bordas de imagens-referência e sua classificação em categorias b_i pré-definidas; 2ª) projeto de máscaras M_i de detecção de bordas b_i baseadas no correspondente perfil médio $\overline{\mathcal{P}_t}$ de níveis de cinza; 3ª) detecção de bordas b_i em novas imagens utilizando a máscara projetada M_i.

Utilizando-se como referência a seqüência de conjuntos ordenados $\mathcal{T}_t$, é imediata a construção dos conjuntos $\mathcal{T}_i^{\mathcal{C}}$, onde as coordenadas dos pixels da imagem segmentada I_{Seg} são substituídas pelos pixels correspondentes da imagem original I_{Orig}. Da formulação 3-23, por exemplo, resulta que:

$$\mathcal{T}_i^{\mathcal{C}} = \left\{ P(x_m, y_m) \in I_{Orig} \cap Obj_i \right\} \qquad (3\text{-}24)$$

Como os traços da microescala podem ser arbitrariamente orientados segundo direção próxima à de um dos eixos horizontal ou vertical da imagem, pode-se construir um detector de bordas unidimensional baseado no perfil médio de variação de níveis de cinza segundo a direção normal a cada um dos traços $\mathcal{T}_i^{\mathcal{C}}$. Se estes, por exemplo,

estiverem aproximadamente alinhados com a direção horizontal da imagem, o seguinte algoritmo, ilustrado através da figura 3.28, produziria uma máscara M_1, específica para a detecção das bordas dos traços mais longos da imagem (T_1):

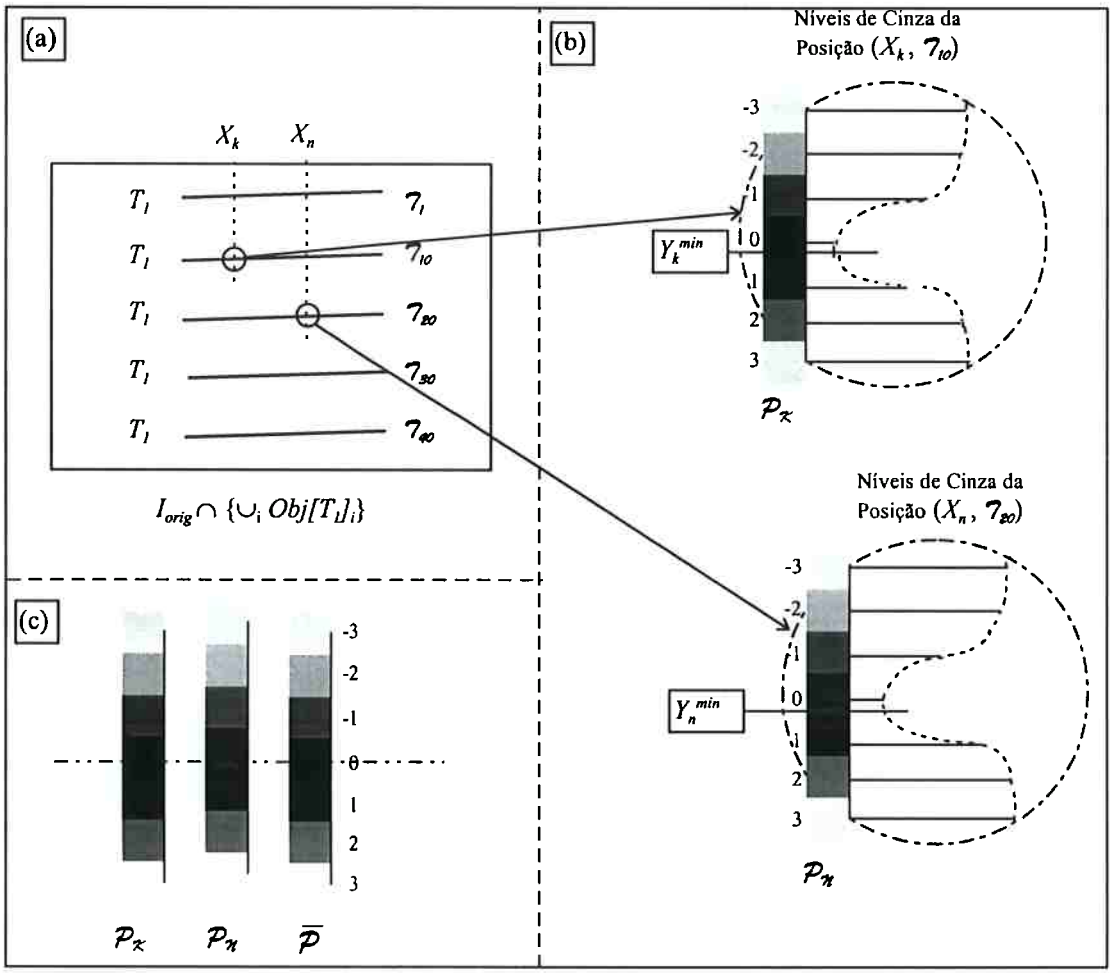

Figura 3.28: Construção de máscara para detecção de bordas de traços do tipo T_1. (a) Imagem original rotulada com os traços do tipo T_1; (b) Perfis de níveis de cinza observados através de janelas verticais V_y; (c) Geração de um perfil médio de níveis de cinza.

- Considerar a imagem I gerada pela intersecção da imagem I_{Orig} (fig. 3.9) com os traços do tipo T_1 da imagem rotulada I_{Rot} (figura 3.27).
- Varrer I com uma janela vertical V_y^0 de tamanho adequado (na figura 3.28, V_y^0 é definida no intervalo [-3,+3]), construindo, para cada nova posição horizontal X_m de V_y^0 sobre um dado traço T_i de I, o perfil $\mathcal{P}_\varkappa$ descrevendo a variação de nível de cinza na direção normal a T_i naquela posição.
- Ajustar uma curva *spline* a cada perfil $\mathcal{P}_\varkappa$ e localizar em V_y^0 a posição Y_m^{min} correspondente ao nível de cinza mínimo de $\mathcal{P}_\varkappa$.

- Para cada perfil $\mathcal{P}_m$ deslocar o centro da janela V_y para o ponto de mínimo Y_m^{min}, obtendo assim a janela V_y^{min}
- Obter os valores interpolados de $\mathcal{P}_m$, observados através da nova janela V_y^{min}, isto é: $\mathcal{P}_m^{int}(-3),\ldots, \mathcal{P}_m^{int}(0),\ldots, \mathcal{P}_m^{int}(+3)$.
- Obter o perfil médio $\overline{\mathcal{P}_l}$ de variação de níveis de cinza da imagem de traços do tipo T_l, efetuando a média aritmética dos valores amostrados de cada um dos perfis $\mathcal{P}_m^{int}$.
- Adotar como máscara M_l de detecção bordas do tipo T_l o perfil médio normalizado $\overline{\mathcal{P}_l}$.

Obviamente, o mesmo procedimento acima descrito para construir a máscara M_1 de detecção de bordas ao longo dos traços do tipo T_1 pode também ser aplicado à construção das máscaras M_2 e M_3, bastando apenas que se utilizem, respectivamente, as imagens $I_{Orig} \cap \{\cup_i Obj[T_2]_i\}$ e $I_{Orig} \cap \{\cup_i Obj[T_3]_i\}$. Alternativamente, pode-se admitir que as características de todos os traços T_1, T_2 e T_3 sejam indistinguíveis, construindo-se uma unica máscara M a partir da imagem $I_{Orig} \cap \{\cup_i Obj[T_1]_i \cup_j Obj[T_2]_j \cup_k Obj[T_3]k_i\}$.

Uma vez construídas as máscaras M_1, M_2 e M_3 (ou eventualmente apenas M), a sua convolução com a imagem original I_{Orig} irá implementar um mecanismo de casamento de padrões, de modo que, para cada abcissa x_m da imagem, bordas tipo *linha* serão localizadas na ordenada y_{Borda} tal que:

$$1.\ P_I(x_m, y_{Borda}) > 0$$

$$2.\ \sum_{\tau=-3}^{\tau=3} I_{Orig}(x_m, y_{Borda}) \cdot M(y_{Borda} - \tau) = max\left\{ \sum_{\tau=-3}^{\tau=3} I_{Orig}(x_m, y) \cdot M(y - \tau) \right\} \qquad (3\text{-}25)$$

onde a condição (1) impõe que $P(x_m, y_{Borda})$ seja um pixel-objeto da imagem I e a condição (2) que, nessa posição, a correlação entre M e I_{Orig} seja máxima.

Na figura 3.29 apresenta-se o resultado da aplicação do algoritmo anterior na localização de bordas do tipo *linha* da imagem I_{Orig} da figura 3.9, usando como referência os traços $\mathcal{T}_i$ rotulados da imagem 3.27.

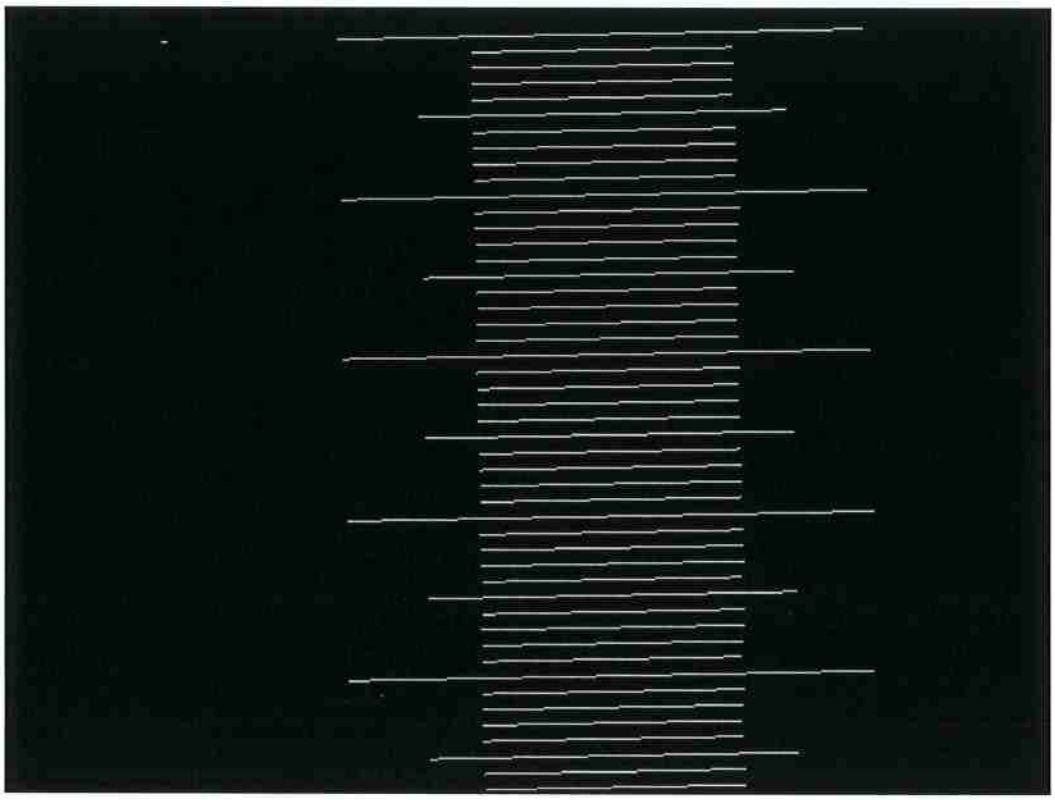

Figura 3.29: Bordas da imagem I_{Orig} (fig. 3.9) detectadas via casamento de padrões com máscaras verticais baseadas nos perfis médios de nível de cinza na direção normal a cada um dos traços $\mathcal{T}_i$.

3.3.1.4 Ferramentas de Medição em Escala Racional

Durante a fase de medição em escala ordinal, usou-se a transformada de Hough para gerar os pares (ρ_i, θ_i) a partir dos quais se construía a série ordenada de conjuntos $\mathcal{T}_i$ representados pela expressão 3-21. Ora, sendo (ρ_i, θ_i) os parâmetros da equação em coordenadas polares da reta $\mathcal{R}_i$ passando através dos pontos do conjunto $\mathcal{T}_i$, poder-se-ia argumentar que, para a estrutura $\mathcal{S}$ de traços paralelos considerada ($\theta_i = \theta$), a simples seqüência ordenada de valores ρ_i já seria o resultado da medição em escala racional. De fato, o processo de medição poderia estar aí concluído não tivessem aquelas estimativas

ρ_i, em geral, um grau de exatidão inferior ao proporcionado pelos métodos de ajuste de retas baseados em mínimos quadrados. Tais métodos, entretanto, são inadequados para identificar precisamente os parâmetros da reta em imagens contendo múltiplas retas, de modo que uma solução de compromisso se faz necessária. Citando-se aqui as próprias palavras de NIBLACK *et* PETKOVIC (1990), "via de regra, a melhor solução de compromisso envolve alguma combinação da transformada de Hough com um método de mínimos quadrados — por exemplo, localizando-se os picos relevantes através da transformada de Hough e, em seguida, ajustando-se retas pelos pontos que contribuem para formar aqueles picos, via mínimos quadrados."

As medidas em escala ordinal anteriormente realizadas, por quaisquer dos dois métodos apresentados, permitem a direta implementação da técnica recomendada acima. Sendo conhecida a seqüência ordenada de n conjuntos $\mathcal{T}_i$, contendo, cada qual, a nuvem de n_i pontos (x_{ij}, y_{ij}), a localização das n retas paralelas $\mathcal{R}_i(\rho_i, \theta)$ pode ser facilmente representada como um problema de minimização de erro médio quadrático. Considere-se, para tanto, a figura 3.30.

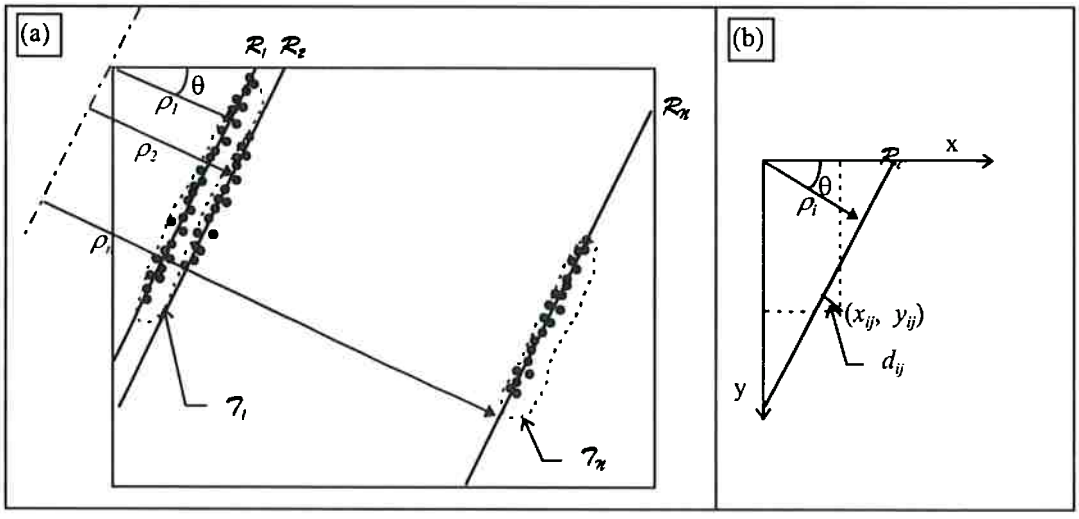

Figura 3.30: Determinação dos parâmetros (ρ_i, θ) das retas.
(a) Seqüências ordenadas $\mathcal{T}_i$; (b) Distância entre ponto P_j de $\mathcal{T}_i$ e reta $\mathcal{R}_i$.

A distância entre um ponto $P_j(x_{ij}, y_{ij})$ do conjunto $\mathcal{T}_i$ à reta ajustada $\mathcal{R}_i$, é:

$$d_{ij} = \left| x_{ij} \cos\theta + y_{ij} \operatorname{sen}\theta - \rho_i \right| \tag{3-26}$$

de modo que o desvio médio quadrático entre as posições dos pontos de $\mathcal{T}_i$ e a sua reta ajustada $\mathcal{R}_i$, é

$$\sigma_i^2 = \frac{1}{n_i} \sum_{j=1}^{n_i} d_{ij}^2 = \frac{1}{n_i} \sum_{j=1}^{n_i} \left(x_{ij} \cos\theta + y_{ij} \operatorname{sen}\theta - \rho_i \right)^2 \tag{3-27}$$

e o desvio médio quadrático total das posições de todos os pontos dos n conjuntos $\mathcal{T}_i$, relativamente às correspondentes retas $\mathcal{R}_i$ é, simplesmente:

$$\sigma^2 = \frac{1}{n_p} \sum_{i=1}^{n} n_i \sigma_i^2 \tag{3-28}$$

$$\text{onde } n_p = \sum_{i=1}^{n} n_i \tag{3-29}$$

Portanto, o mínimo desvio médio quadrático total irá corresponder às posições (ρ_i, θ) das retas $\mathcal{R}_i$, tais que:

$$\frac{\partial \sigma^2}{\partial \theta} = 0 \tag{3-30}$$

$$\frac{\partial \sigma^2}{\partial \rho_i} = 0 \quad i = 1, \ldots, n \tag{3-31}$$

Expandindo e desenvolvendo o lado esquerdo da equação 3-30, resulta:

$$\frac{1}{2n_p} \frac{\partial \sigma^2}{\partial \theta} = \mathcal{A} \cos\theta \operatorname{sen}\theta + \mathcal{B}\left(\cos^2\theta - \operatorname{sen}^2\theta \right) + \left(\sum_{i=1}^{n} \mathcal{C}_i \rho_i \right) \operatorname{sen}\theta - \left(\sum_{i=1}^{n} \mathcal{D}_i \rho_i \right) \cos\theta \tag{3-32}$$

onde:

$$\mathcal{A} = \sum_{i=1}^{n} \sum_{j=1}^{n_i} \left(y_{ij}^2 - x_{ij}^2 \right) \tag{3-33}$$

$$\mathcal{B} = \sum_{i=1}^{n}\sum_{j=1}^{n_i} x_{ij}\, y_{ij} \tag{3-34}$$

$$\mathcal{C}_i = \sum_{j=1}^{n_i} x_{ij} \tag{3-35}$$

$$\mathcal{D}_i = \sum_{j=1}^{n_i} y_{ij} \tag{3-36}$$

Fazendo o mesmo em relação à equação 3-31, segue que:

$$\frac{1}{2n_p}\frac{\partial E^2}{\partial \rho_i} = \mathcal{C}_i \cos\theta + \mathcal{D}_i \operatorname{sen}\theta - n_i \rho_i \tag{3-37}$$

Portanto, de 3-31 e 3-37, resulta que:

$$\rho_i = \frac{1}{n_i}\left(\mathcal{C}_i \cos\theta + \mathcal{D}_i \operatorname{sen}\theta\right) \tag{3-38}$$

Substituindo 3-38 em 3-32, e igualando-se a expressão resultante a 0, obtém-se:

$$\mathcal{A}\frac{\operatorname{sen} 2\theta}{2} + \mathcal{B}\cos 2\theta + \left(\sum_{i=1}^{n}\frac{\mathcal{C}_i^2}{n_i} - \sum_{i=1}^{n}\frac{\mathcal{D}_i^2}{n_i}\right)\frac{\operatorname{sen} 2\theta}{2} - \left(\sum_{i=1}^{n}\frac{\mathcal{C}_i\mathcal{D}_i}{n_i}\right)\cos 2\theta = 0 \tag{3-39}$$

Usando a notação

$$\mathcal{G} = \sum_{i=1}^{n}\frac{\mathcal{C}_i^2}{n_i} \tag{3-40}$$

$$\mathcal{K} = \sum_{i=1}^{n}\frac{\mathcal{D}_i^2}{n_i} \tag{3-41}$$

$$\mathcal{L} = \sum_{i=1}^{n}\frac{\mathcal{C}_i\mathcal{D}_i}{n_i} \tag{3-42}$$

a equação 3-39 permite determinar o ângulo θ, como

$$\theta = \frac{1}{2}\tan^{-1}\left(2 \times \frac{\mathcal{L} - \mathcal{B}}{\mathcal{A} + \mathcal{K} - \mathcal{G}}\right) \tag{3-43}$$

de modo que as medidas ρ_i, descrevendo as posições relativas das retas $\mathcal{R}_i$ da imagem da microescala são, finalmente, determinadas substituindo-se o valor θ acima na equação 3-38, enquanto que as correspondentes incertezas σ_i dessas medidas são imediatamente calculadas a partir da expressão 3-27. Aqui, porém, cabe um parênteses importante acerca do cálculo da incerteza dos parâmetros da equação de uma reta localizada em imagens digitais. Conforme demonstrado por (CAGLIOTI, 1993), quando a imagem da reta se aproxima de um segmento de linha digital (*SLD*)[10], os valores σ_i de incerteza calculados pela formulação baseada em modelo de reta contínua (equação 3-27) são, na verdade, sub-estimativas das incertezas reais que afetam as medidas σ_i, já que esse modelo não considera os efeitos da quantização da imagem.

A unidade das medidas ρ_i estimadas pelo processo anterior, é o *nº de pixels*. Todavia, conforme já mencionado no início deste capítulo, em ambas as categorias de medição, sejam *rotineiras* (ensaios metrológicos), sejam *criativas* (planejamento e construção dos instrumentos metrológicos), as medidas devem ser expressas em unidades métricas. No primeiro caso, por motivo óbvio; no segundo, porque a avaliação da qualidade dos instrumentos de visão computacional projetados utilizará um conjunto de medidas-referência obtidas de ensaio metrológico convencional, expressas, portanto, em unidades métricas. A calibração daqueles instrumentos, através da determinação de um fator de escala k que converta *nº de pixels* em mm, pode ser realizada de duas formas distintas: 1ª) operando-se com uma única imagem e adotando-se as medidas do ensaio convencional como referência; 2ª) operando-se sobre um par de imagens da microescala deslocada mecanicamente de uma distância D segundo a direção $\vec{n}$ (horizontal ou vertical) e utilizando-se como referência a sua projeção na direção normal às retas.

---

[10] Em inglês, usualmente abreviado como DLS, de *Digital Line Segment.*

Considere-se, então, para o primeiro caso, $\boldsymbol{r}$ como o vetor das posições r_i dos traços da microescala determinadas pelo método convencional, e ρ como o vetor das posições ρ_i das correspondentes retas na imagem. Para o segundo caso seja $\boldsymbol{r}$ o vetor $\vec{D}$ projetado na direção normal às retas da imagem e ρ o vetor de distâncias entre retas correspondentes nas imagens deslocadas. Admitindo-se, em ambos os caso, $\boldsymbol{r}$ como um vetor determinístico de medidas-referência e ρ como o vetor de observações ρ_i geradas pelo instrumento de visão computacional, o fator de escala k, convertendo *nº de pixels* em unidades métricas, pode ser obtido através da formulação derivada do teorema fundamental de Gauss-Markov para o caso particular de relação linear entre os vetores de observações ρ e de estado $\boldsymbol{k}$, não-correlação entre estado e ruído e desconhecimento do estado, ou seja:

$$\boldsymbol{k} = k = \left(\boldsymbol{r}^T \boldsymbol{C}_v^{-1} \boldsymbol{r}\right)^{-1} \boldsymbol{r}^T \boldsymbol{C}_v^{-1} \rho \tag{3-44}$$

onde são conhecidos os vetores de medidas-referência $\boldsymbol{r}$, de observações ρ e a matriz $\boldsymbol{C}_v$ de covariância de ruído das observações:

$$\boldsymbol{r} = \begin{bmatrix} r_1 & r_2 & \cdots & r_n \end{bmatrix}^T \tag{3-45}$$

$$\rho = \begin{bmatrix} \rho_1 & \rho_2 & \cdots & \rho_n \end{bmatrix}^T \tag{3-46}$$

$$\boldsymbol{C}_v = \begin{bmatrix} \sigma_1^{-2} & & & \\ & \sigma_2^{-2} & & \\ & & \ddots & \\ & & & \sigma_n^{-2} \end{bmatrix} \tag{3-47}$$

Expandindo a expressão 3-44, resulta que o fator de escala k é dado por:

$$k = \frac{\sum_{i=1}^{n} \frac{r_i \rho_i}{\sigma_i^2}}{\sum_{i=1}^{n} \frac{r_i^2}{\sigma_i^2}} \tag{3-48}$$

o que permite a imediata conversão de *nº de pixels* em unidades métricas.

3.3.2 Ferramentas Emulando a Medição com Gabaritos

De acordo com a norma brasileira de peneiras para ensaio ABNT-5734 (ABNT, 1991), a calibração desses padrões pode ser realizada alternativamente por dois métodos distintos de medição: deslocamento mecânico monitorado por imagem ou comparação com gabaritos-referência.

No primeiro caso, conforme já relatado, a imagem da tela, ampliada por projetor de perfis (figura 1.2), é utilizada pelo operador como guia para ações de deslocamento mecânico, através das quais são obtidos os valores médios da abertura a_T da tela e do diâmetro d_f dos fios (figura 3.31-a). Em particular, a mesma norma determina que sejam escolhidas, aleatoriamente, n "áreas de observação", sobre as quais se realizam um número mínimo m de medições ao longo de duas direções diagonais perpendiculares da tela (figura 3.31-b).

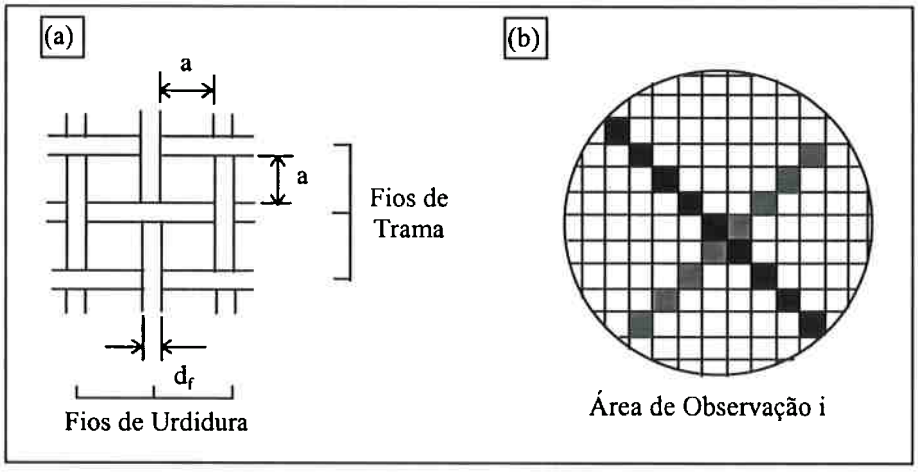

Figura 3.31: (a) Dimensões características da tela da peneira; (b) Zonas de medição.

Quanto à segunda técnica de medição referida, esta se baseia na utilização de esferas-padrão de vidro de diâmetro nominal d_e, com distribuição granulométrica

conhecida e certificada [11]. Através de pesagem, determina-se o percentual pc_T de esferas passantes através da peneira-teste, com abertura a_T. De acordo com o certificado do padrão d_e, sendo pc_1 e pc_2 ($pc_1 < pc_T < pc_2$) os percentuais de esferas passantes através de peneiras com aberturas nominais a_1 e a_2, respectivamente, a abertura efetiva da peneira-teste é obtida através de simples interpolação linear:

$$a_T = a_1 + \frac{pc_T - pc_1}{pc_2 - pc_1}(a_2 - a_1) \tag{3-49}$$

Esta última técnica de medição pode ser vista em termos genéricos como uma atividade de inspeção de tolerâncias, a qual, segundo GHOSH (1990), corresponde ao conhecido *Problema de Inclusão do Polígono*, qual seja: "verificar se um polígono P pode ser contido e transladado no interior de outro polígono Q e, em caso positivo, identificar a região R em que essa translação é possível".

Utilizando-se a linguagem da morfologia matemática, o referido problema, para o caso específico de inspeção de tolerâncias das aberturas da tela da peneira, seria assim enunciado: "Seja T a imagem binária da tela de uma peneira de ensaio P, onde as aberturas retangulares correspondem a n_T objetos A_i rotulados. Sejam t_{min} e t_{max} as tolerâncias mínima e máxima admissíveis para a abertura nominal a_n e sejam C_{min} e C_{max} elementos estruturantes *círculo* com diâmetros, respectivamente, d_{min} e d_{max} — inteiros, ímpares e expressos em *nº de pixels* — correspondendo às aberturas a_n-t_{min} e a_n+t_{max}:

$$npix_{min} = \left\lceil (a_n - t_{min}) \times k \right\rceil \tag{3-50}$$

$$npix_{max} = \left\lceil (a_n - t_{max}) \times k \right\rceil \tag{3-51}$$

onde k é o fator de escala (*nº de pixels/mm*). Determinar as percentagens pc_1 e pc_2 de objetos A_i com aberturas a tais que: $a > a_n + t_{max}$ e $a < a_n - t_{min}$."

[11] Essas esferas são fornecidas pelo NIST — National Institute of Standards and Technology.

A solução desse problema é imediata, quando se aplicam à imagem binária T operações de erosão convenientes. Assim, erodindo-se T com o elemento estruturante $C_{max}(d_{max}+1)$, resulta a imagem R_M dos resíduos das aberturas maiores que a_n+t_{max}:

$$R_M = T\Theta C_{max}\left(d_{max}+1\right) \tag{3-52}$$

de modo que a percentagem pc_1 dessas aberturas é dada por

$$pc_1 = \frac{n\left(R_M\right)}{n_T} \tag{3-53}$$

onde $n(R_M)$ é o numero de objetos da imagem rotulada de R_M. Analogamente, a percentagem pc_2 de aberturas inferiores a a_n-t_{min} é:

$$pc_2 = \frac{n_T - n\left(R_m\right)}{n_T} \tag{3-54}$$

onde R_m é a imagem-resíduo de T após erosão com o elemento estruturante $C_{min}(d_{min}$-1):

$$R_m = T\Theta C_{min}\left(d_{min}-1\right) \tag{3-55}$$

Considere-se a figura 3.32, onde se apresenta a imagem parcial T de uma tela de peneira padrão ABNT-325, com 45 μm de abertura nominal, previamente calibrada pelo método convencional, baseado em projetor de perfis. A implementação do método morfológico de medição anteriormente esboçado requer as seguintes operações preliminares: a) T_L = T limiarizada; b) $T_L^R = T_L$-OI, onde OI são os objetos incompletos que tocam as fronteiras de T_L; c) $T_{Rot} = T_L^R$ rotulada; d) $n_T = n(T_{Rot})$ = número de aberturas de T. Na imagem da figura 3.33 apresentam-se as 231 aberturas rotuladas da imagem T da figura 3.32, após limiarização usando limiar estimado pelo método de HADDON (1988) e eliminação dos objetos incompletos OI identificados através da seguinte operação morfológica:

$$OI = Moldura\left(T_L\right)\oplus|_{T_L}\ C(r=1) \tag{3-56}$$

onde *Moldura*(T_L) é a imagem do retângulo limítrofe da imagem T_L e $\oplus|_T C(r=1)$ é a operação de *dilatação condicional* (HARALICK; SHAPIRO,1991) com o círculo de raio unitário relativamente à imagem limiarizada T_L.

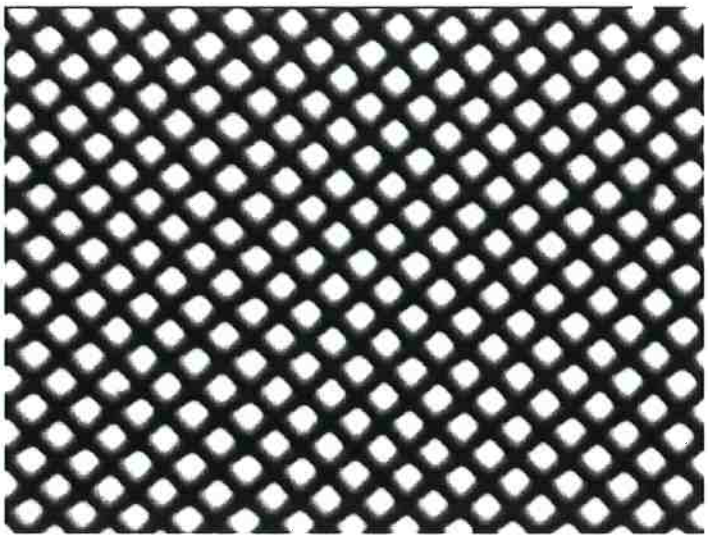

Figura 3.32: Imagem da tela de uma peneira padrão ABNT-325.

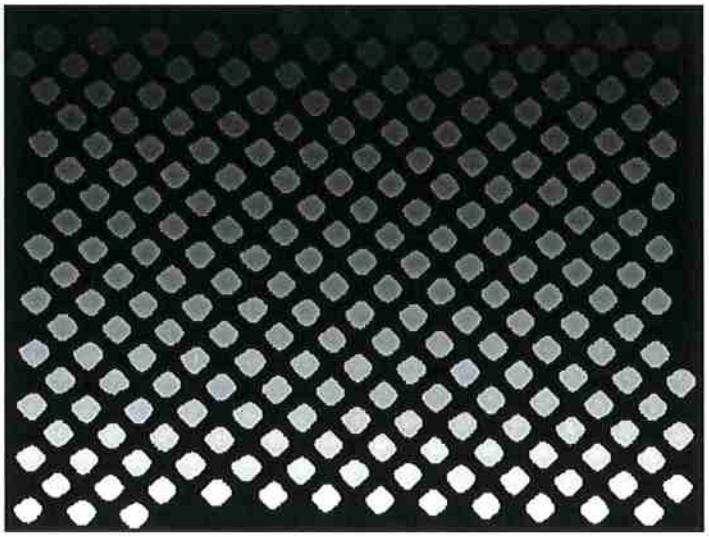

Figura 3.33: Limiarização, regularização e rotulação da imagem da figura 3.32.

Após esse processamento preliminar, verificam-se as tolerâncias das aberturas da peneira simplesmente aplicando-se as operações 3-52 a 3-55 acima. Para isto, porém, é necessário que se conheça *a priori* o valor do fator de escala *k*, ou dos fatores de escala

k_A, k_B em duas direções ortogonais $\vec{n}_A, \vec{n}_B$, caso os pixels da imagem não sejam exatamente quadrados. Algoritmos de visão computacional baseados no método evolutivo, análogos aos utilizados nas imagens de microescalas, podem ser aplicados para se obter esses fatores, feitas as seguintes considerações:

1. O resultado da medição em escala ordinal, via transformada de Hough, são dois conjuntos ordenados $\mathcal{T}_A$ e $\mathcal{T}_B$ de retas aproximadamente ortogonais (figura 3.34). Porém, como pares de retas sucessivas [$\mathcal{R}_i$, $\mathcal{R}_{i+1}$] podem corresponder ora a um fio ora a aberturas da trama, é necessário classificar $\mathcal{T}_A$ e $\mathcal{T}_B$ como seqüências dos tipos $\mathcal{S}_1[T_{A\text{-}F} \rightarrow T_{F\text{-}A} \rightarrow T_{A\text{-}F} \rightarrow \ldots]$ ou $\mathcal{S}_2[T_{F\text{-}A} \rightarrow T_{A\text{-}F} \rightarrow T_{F\text{-}A} \rightarrow \ldots]$, onde $T_{A\text{-}F}$ representa a interface *abertura→fio* e $T_{F\text{-}A}$, *fio→abertura*.
2. Para cada um dos conjuntos ordenados $\mathcal{T}_A$ e $\mathcal{T}_B$, a medição em escala racional é realizada de forma análoga à apresentada no ítem 3.3.1.4: os ângulos θ_A e θ_B são determinados pela equação 3-43, os vetores ρ_A e ρ_B, através da equação 3-38 e os fatores de escala k_A e k_B, através da equação 3-48. Em relação a esses fatores, quando se utilizam medidas-referência, ρ_A e ρ_B correspondem à distância (em *nº de pixels*) entre duas retas adjacentes descritoras de um fio e $r_A=r_B=r$ corresponde ao valor médio d_f do diâmetro do fio, obtido através do processo tradicional. Utilizando-se, alternativamente, o processo de calibração por deslocamento, ρ_A e ρ_B são as distâncias entre retas correspondentes nas imagens da microtextura deslocada mecanicamente de uma distância conhecida D, segundo um dos eixos x ou y, de modo que r_A, r_B correspondem, nesse caso, a esse valor D projetado segundo as direções $\vec{n}_A, \vec{n}_B$.

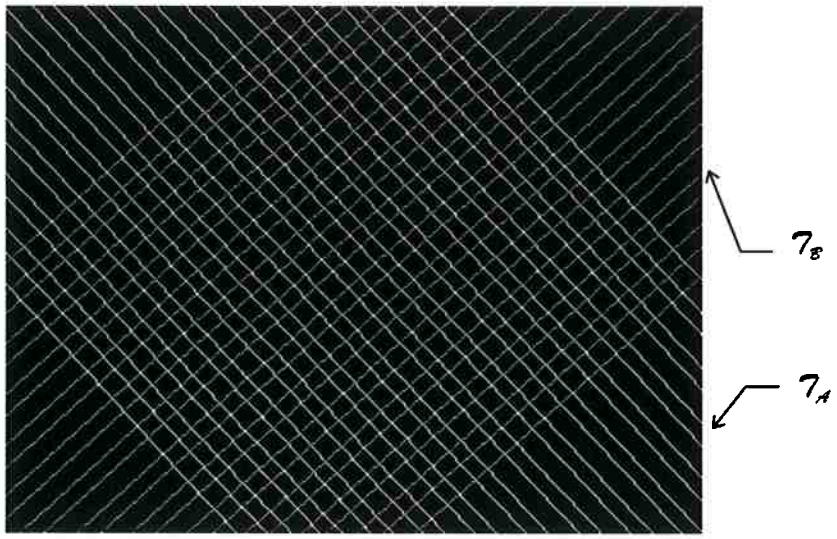

Figura 3.34: Conjuntos ordenados de retas localizadas na imagem 3.32.

Considerando-se o caso mais geral, sendo conhecidos os fatores k_A e k_B e os ângulos θ_A, θ_B, constroem-se, em lugar dos elementos estruturantes C_{min} e C_{max} do tipo *Círculo*(d), elementos estruturantes E_{min} e E_{max} do tipo *Elipse*(d_A,d_B), utilizando-se as correspondentes fórmulas 3-50 e 3-51 para o cálculo dos diâmetros principais e orientando-se estes segundo as direções definidas pelos ângulos θ_A, θ_B. Para aumentar a eficiência dos algoritmos morfológicos é conveniente ainda simplificar os elementos estruturantes do tipo *Elipse* substituindo-os por conjuntos da forma

$$\tilde{E}\left(d_A, d_B, \vec{n}_A, \vec{n}_B\right) = \left\{(0,0), \left(0, \frac{d_A}{2}\right)\vec{n}_A, \left(0, -\frac{d_A}{2}\right)\vec{n}_A, \left(\frac{d_B}{2}, 0\right)\vec{n}_B, \left(\frac{d_B}{2}, 0\right)\vec{n}_B\right\} \qquad (3\text{-}57)$$

A imagem da figura 3.35 corresponde aos objetos residuais R_m^i resultantes da erosão da imagem da figura 3.33 pelo elemento estruturante $\tilde{E}_{min}\left(d_A^{min}, d_B^{min}, \vec{n}_A, \vec{n}_B\right)$. Sendo o número de objetos R_m^i igual ao da imagem original T, conclui-se que nenhuma das aberturas da peneira, na região considerada, é inferior a $a_N - t_{min}$.

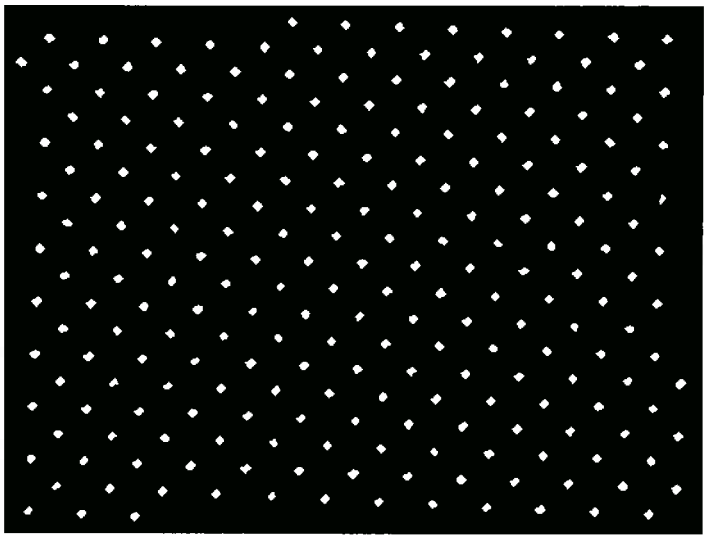

Figura 3.35: Inspeção de tolerância das aberturas mínimas na imagem da figura 3.33.

3.3.3 Ferramentas de Medição Puramente Computacionais

Para certas categorias de imagens, representando cenas com estruturas bastante simples, uma possível abordagem consiste na aplicação direta de algoritmos de medição em escala racional, sem qualquer processamento prévio da imagem. Ora, isto exatamente ocorre nos casos aqui ilustrados, em que ambas as imagens, da microescala e da tela da peneira, podem ser representadas por padrões repetitivos de retas paralelas, dos quais se podem estimar as taxas de repetição e as posições individuais das retas, respectivamente, através de métodos de estimação de freqüências espaciais e da transformada de Hough.

O método desenvolvido por KHALAJ *et al* (1994), originalmente aplicado a imagens de máscaras de circuito integrado, estima freqüências espaciais de estruturas repetitivas — no caso, seqüências de objetos retangulares —, perfeitamente alinhadas em relação aos eixos da imagem (figura 3.36). Nessas condições, o problema de estimação se resume à identificação das freqüências fundamentais das ondas quadradas idealizadas que melhor representam os sinais

$$x_i = \sum_{j=1}^{n_y} \left(\vec{L}_j\right)_i \qquad i = 1 \ldots n_x \tag{3-58}$$

$$y_i = \sum_{j=1}^{n_x} \left(\vec{L}_i\right)_j \qquad i = 1 \ldots n_y \tag{3-59}$$

correspondentes, respectivamente, as projeções das linhas $\vec{L}_i$ da imagem I

$$I = \begin{bmatrix} \vec{L}_1 \\ \vdots \\ \vec{L}_{n_y} \end{bmatrix} \tag{3-60}$$

segundo as direções y (colunas) e x (linhas).

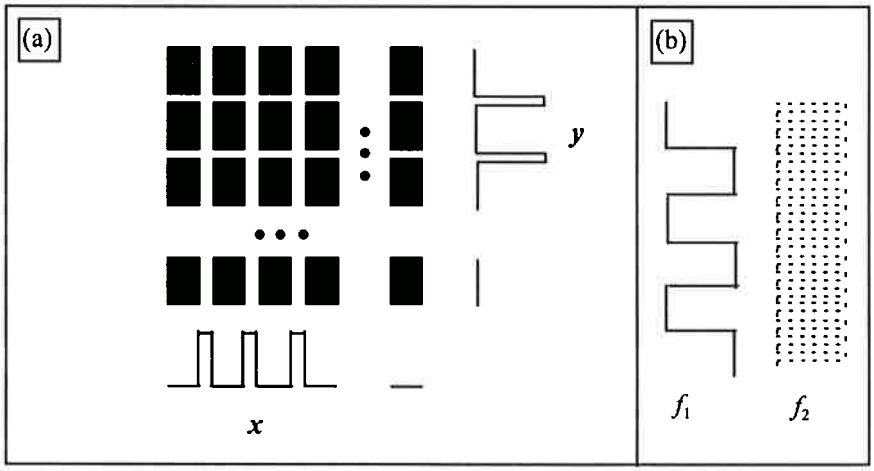

Figura 3.36: (a)Projeções da imagem de padrões repetitivos; (b) Freqüências principais do sinal y.

Este problema é solucionado representando-se, inicialmente, o vetor $\boldsymbol{x}$, de dimensão n_x, (e $\boldsymbol{y}$, analogamente) como um vetor $\boldsymbol{X}$ de sub-vetores $\boldsymbol{p}_i$ de dimensão m, ($2 < m \leq n_x$-1) na forma

$$\boldsymbol{X} = \begin{bmatrix} x_1 & x_2 & \cdots & x_{n_x-m+1} \\ x_2 & x_3 & \cdots & x_{n_x-m+2} \\ \vdots & \vdots & \vdots & \vdots \\ x_m & x_{m+1} & \cdots & x_{n_x} \\ \boldsymbol{p}_1 & \boldsymbol{p}_2 & \cdots & \boldsymbol{p}_{n_x} \end{bmatrix} \tag{3-61}$$

e identificando-se, através do algoritmo TLS-ESPRIT [12], descrito em (ROY; KAILATH, 1989), o sub-espaço $\boldsymbol{E}_s$ de dimensão 4 (referente aos dois pares de freqüências, positiva e negativa, da onda quadrada idealizada) que melhor se aproxima do espaço $\boldsymbol{X}$, ou seja,

$$\boldsymbol{E}_s = \begin{bmatrix} \boldsymbol{e}_1 & \boldsymbol{e}_2 & \boldsymbol{e}_3 & \boldsymbol{e}_4 \end{bmatrix}, \tag{3-62}$$

onde $\boldsymbol{e}_i$ (i=1,...,4) são os quatro autovetores de maior módulo da matriz $\boldsymbol{R}_{xx}$ de autocovariância de $\boldsymbol{X}$:

$$\boldsymbol{R}_{xx} = \frac{1}{n_x - m + 1} \boldsymbol{X} \cdot \boldsymbol{X}^T \tag{3-63}$$

Conforme apresentado em (KHALAJ *et al*, 1994), as duas freqüências espaciais principais do sinal $\boldsymbol{x}$ são dadas por:

$$f_k = \frac{1}{2\pi} \tan^{-1} \frac{\mathrm{Im}(\zeta_k)}{\mathrm{Re}(\zeta_k)} \qquad k = 1,\ldots,2 \tag{3-64}$$

onde ζ_k é o $k^{\text{ésimo}}$ autovalor da matriz $[-\boldsymbol{F}_{12}\boldsymbol{F}_{22}^{-1}]$ e $\boldsymbol{F}_{12}$ e $\boldsymbol{F}_{22}$ são partições de

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{F}_{11} & \boldsymbol{F}_{12} \\ \boldsymbol{F}_{21} & \boldsymbol{F}_{22} \end{bmatrix}, \tag{3-65}$$

matriz de autovetores de

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{E}_1 \\ \boldsymbol{E}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{E}_1 & \boldsymbol{E}_2 \end{bmatrix}, \tag{3-66}$$

em que $\boldsymbol{E}_1$ contém as linhas 1 a m-1 da matriz $\boldsymbol{E}_s$ e $\boldsymbol{E}_2$, as linhas 2 a m da mesma matriz:

$$\boldsymbol{E}_1 = \begin{bmatrix} e_{11} & \cdots & e_{41} \\ \vdots & \vdots & \vdots \\ e_{1m-1} & \cdots & e_{4m-1} \end{bmatrix}$$
$$\boldsymbol{E}_2 = \begin{bmatrix} e_{12} & \cdots & e_{42} \\ \vdots & \cdots & \vdots \\ e_{1m} & \cdots & e_{4m} \end{bmatrix} \tag{3-67}$$

[12] Total Least Square Estimation of Signal Parameters via Rotational Invariance Technique.

Na figura 3.37 apresenta-se o resultado da aplicação desse algoritmo à imagem da microescala, estando os seus traços devidamente alinhados com o eixo horizontal. A título de ilustração, as retas uniformemente espaçadas, correspondentes a um período espacial fundamental de 8,62 pixels, foram superpostas à imagem original, assumindo-se conhecida a posição da primeira reta do feixe.

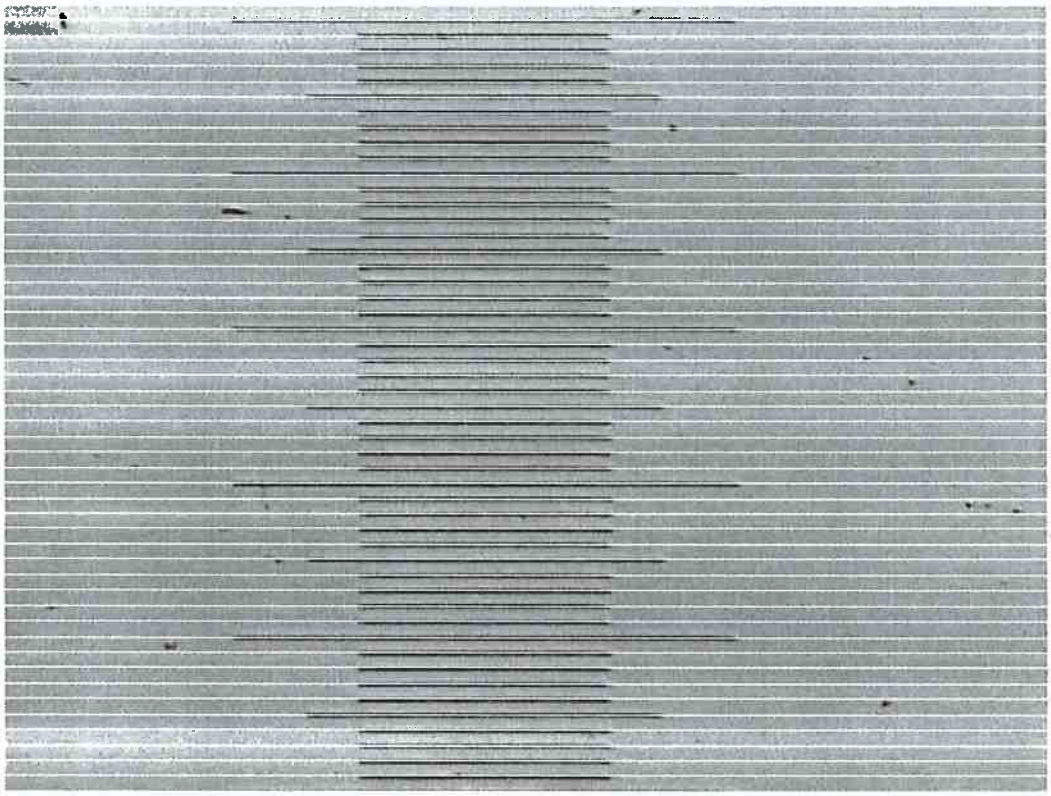

Figura 3.37: Retas localizadas via estimação de freqüências espaciais.

PETKOVIC *et al* (1988) propõem uma versão da transformada de Hough que não requer qualquer pré-processamento da imagem original I: o espaço discreto $\mathcal{H}(\rho \times \theta)$, nesse caso, é o resultado da acumulação dos níveis de cinza dos pixels da imagem I projetados em cada posição (ρ, θ), e não mais do número de pixels da imagem binária de bordas de I. Com isso, o método de localização de retas se inverte completamente: tradicionalmente, retas correspondem a máximos locais de $\mathcal{H}(\rho \times \theta)$, função acumulação

do número de máximos locais do gradiente [13] de níveis de cinza de I; no método alternativo, retas são pontos de máximo local do gradiente de $\mathcal{H}(\rho \times \theta)$, função acumulação dos níveis de cinza de I. Conforme salientado por aqueles autores, esse processo é equivalente à detecção de bordas utilizando-se operadores locais com comprimento equivalente ao das retas da imagem; sua implementação, porém, é muito mais eficiente do que a construção de famílias de máscaras longas de detecção, uma para cada particular ângulo θ considerado.

Na figura 3.38 apresenta-se a imagem de retas detectadas na imagem da figura 3.9 (superposta a esta), através do algoritmo baseado na transformada de Hough, proposto por PETKOVIC *et al* (1988).

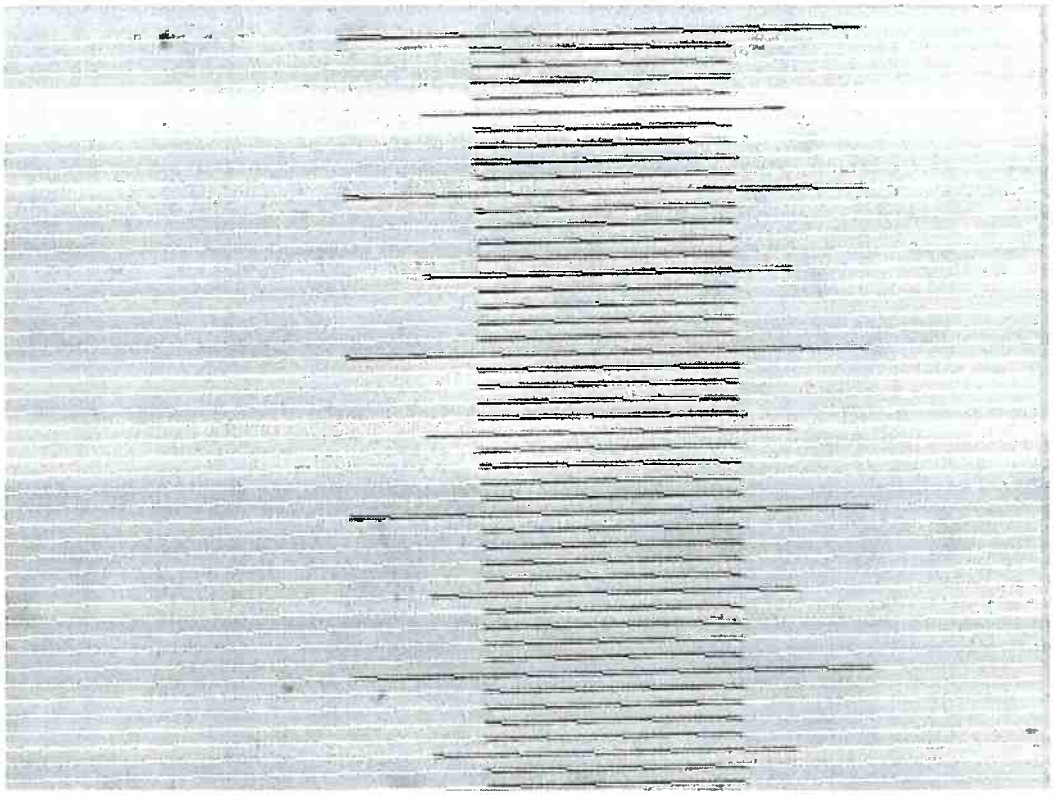

Figura 3.38: Retas da imagem da figura 3.9 localizadas pelo algoritmo de PETKOVIC et al (1988).

[13] Ou "zero crossings" se se utilizar detector de bordas baseado no Laplaciano da imagem.

4 - INSTRUMENTOS METROLÓGICOS DE VISÃO COMPUTACIONAL

Com o intuito de automatizar os ensaios rotineiros executados pelo Laboratório de Metrologia do IPT, para calibração de microescalas e de microtexturas de peneiras de medição, foram construídos instrumentos metrológicos inteligentes utilizando-se uma mesma plataforma de aquisição de imagens e diversas combinações alternativas de algoritmos de visão computacional, organizados segundo os princípios apresentados no capítulo anterior. Para a avaliação do desempenho dos instrumentos assim constituidos, associaram-se medidas-referência de microescalas e de microtexturas, obtidas através de procedimentos metrológicos tradicionais.

4.1 Plataforma de Aquisição de Imagens

O sistema de aquisição de imagens foi configurado utilizando-se equipamentos e recursos que já faziam parte do instrumental do Laboratório de Metrologia do IPT, ou que foram especificados muito antes de se dar início ao desenvolvimento do projeto de automatização laboratorial. Os ítens selecionados incluem:

- Microscópio de Medição: Utilizou-se um microscópio marca Leitz modelo CMM-2D ("2D Component Measuring Microscope"), com sistemas de iluminação episcópica e diascópica [1], mesa de coordenadas micrométricas (50mmx50mm), três oculares, sendo uma para captura de imagem através de câmera de vídeo, e quatro objetivas (3,2x, 10x, 20x, 50x), totalizando um poder de resolução de até 1500x.

[1] Respectivamente, por transmissão e por reflexão.

- Tambores Micrométricos: À mesa do microscópio foram acoplados dois tambores micrométricos marca Mitutoyo modelo Digimatic-164, para medição e registro digital de translações X-X e Y-Y de até 50 mm, com resolução de 1,0 μm.
- Sistema Laser de Medição: Utilizou-se o sistema laser Renishaw ML-10, para a medição precisa (resolução de 0,1μm) de deslocamentos X-X ou Y-Y da mesa do microscópio, necessários à determinação do fator de conversão *k* (*nº de pixels*/mm) utilizado na calibração da microescala.
- Câmera de Vídeo CCD: Ao tubo do microscópio acoplou-se uma câmera de vídeo marca Optronics, modelo LE-470, baseada em um sensor CCD Sony 1/2", com área sensível de 6,4mm x 4,8mm e arquitetura de transferência entrelaçada de linhas, gerando na saída sinal de vídeo composto, com padrão RS-170.
- Sistema de Digitalização de Imagens: À saída da câmera acoplou-se o sistema de digitalização de imagens MuTech M-Vision 1000 (MUTECH, 1995), com entradas para até quatro sinais de vídeo, circuito identificador de sinais separando os sinais de sincronismo do sinal de vídeo, circuito de condicionamento do sinal de vídeo e circuito PLL capaz de gerar sinal temporizador com *jitter* inferior a 10*ns*.

A quase totalidade dos recursos acima referidos se encontram ao nível de estado da arte para o tipo de aplicação pretendida. O microscópio de medição escolhido possui um sistema de iluminação de campo uniforme e um sistema de lentes que realiza projeção paralela, produzindo imagens uniformemente iluminadas e com distorção virtualmente nula. O sistema de digitalização de imagens, bastante avançado, possui, inclusive, recursos para sincronização da freqüência de digitalização com o sinal de temporização de pixel da câmera CCD. Esta última, contudo, por utilizar o método de transferência entrelaçada de linhas e não o de transferência quadro a quadro, não se constitui em escolha ideal para aplicações metrológicas, conforme apresentado no

capítulo anterior; no entanto, teve que ser utilizada porque os recursos financeiros disponíveis para o projeto de automatização não eram suficientes para a aquisição de novos equipamentos.

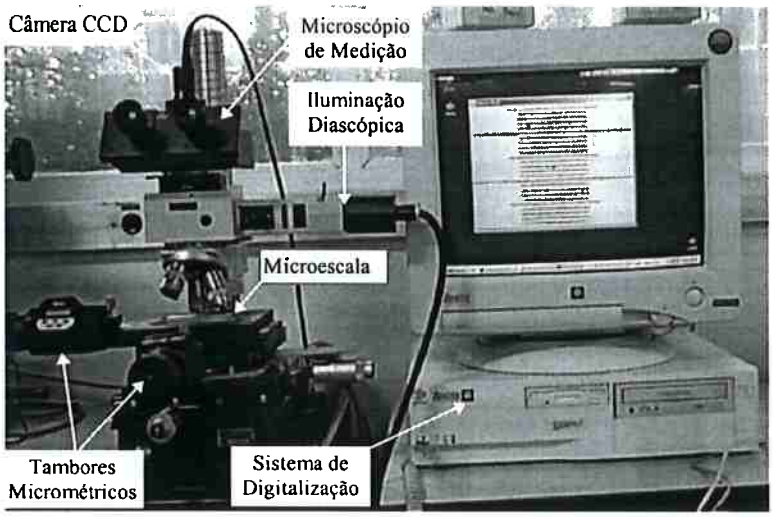

Figura 4.1: Arranjo experimental para captura de imagens de microescalas.

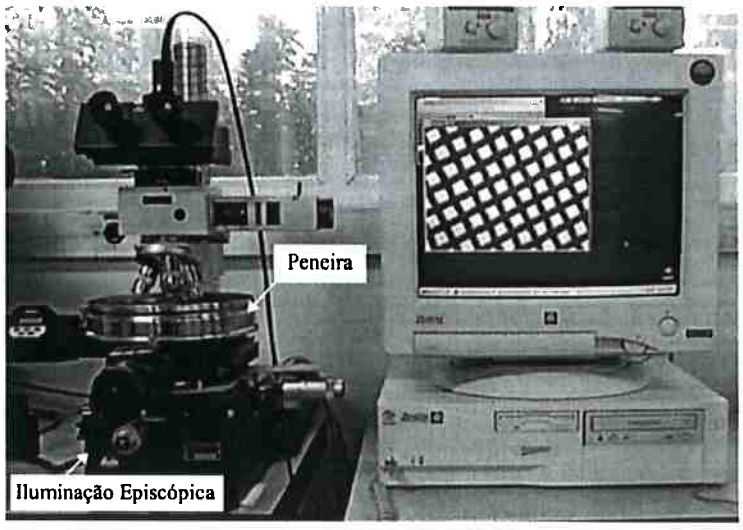

Figura 4.2: Arranjo experimental para a captura de imagens de microtexturas de peneiras.

Nas figuras 4.1 e 4.2 apresentam-se, respectivamente, os arranjos experimentais para a captura de imagens de microescalas e da microtextura de peneiras de medição, a diferença entre ambos residindo basicamente na forma de iluminação — diascópica, no primeiro caso e episcópica no segundo.

4.2 Medidas-Referência

Para avaliar a exatidão e repetibilidade das medidas geradas pelos instrumentos metrológicos computacionais construídos, utilizaram-se os resultados apresentados em certificados emitidos pelo Laboratório de Metrologia do IPT, referentes à calibração de uma microescala de 1mm e uma peneira padrão ABNT-325 (abertura nominal= 45μm).[2]

No caso da microescala, a imagem-referência utilizada no ensaio convencional apresentava a máxima amplificação permitida pelo microscópio de medição (objetiva de 50x) e as posições dos traços foram determinadas deslocando-se manualmente a mesa micrométrica do microscópio e capturando-se as suas coordenadas, com auxílio de um sistema de medição laser, sempre que se observava no monitor o alinhamento das bordas de um novo traço da microescala com o par fixo de linhas paralelas do cursor-referência. A instrumentação utilizada nesse ensaio, conforme se indica na figura 4.3, inclui: microscópio de medição Leitz CMM-2D, câmera de vídeo Minilux Super CCTV, e sistema de medição laser Renishaw ML-10.

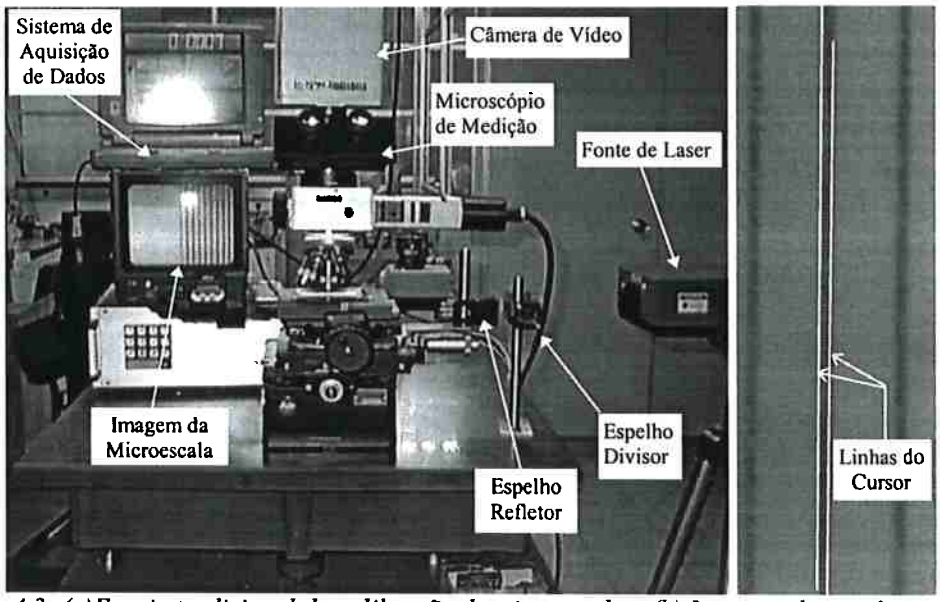

Figura 4.3: (a)Ensaio tradicional de calibração de microescalas; (b) Imagem observada no ensaio.

[2] Posteriormente, quando estes se mostraram menos precisos que os instrumentos computacionais construídos, efetuou-se a calibração baseada em deslocamento, conforme se apresenta no capítulo 5.

No caso da peneira, utilizou-se também a máxima amplificação (100x) permitida pelo projetor de perfis Jones & Lamson modelo Epic (figura 4.4), obtendo-se os valores médios da abertura e do diâmetro dos fios, bem como suas respectivas tolerâncias, de acordo com procedimento baseado na norma ASTM-E11-87 (ASTM, 1987).

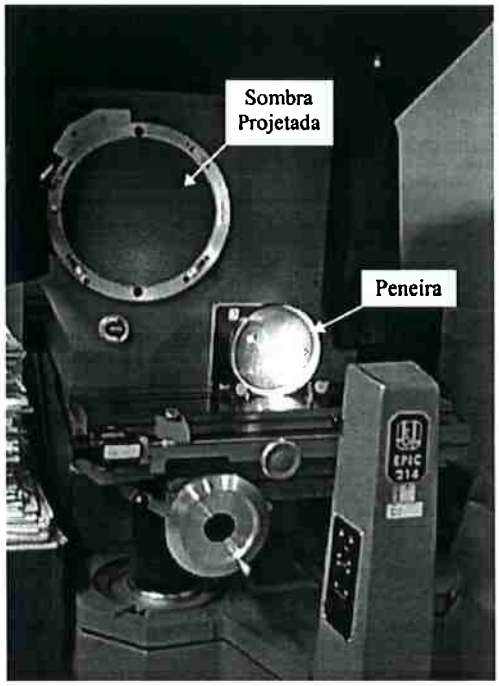

Figura 4.4: Ensaio de calibração de microtextura de peneira de medição.

Nas tabelas 4-1 e 4-2 apresentam-se, respectivamente, as medidas-referência utilizadas para a avaliação da exatidão das medidas produzidas pelos instrumentos computacionais de medição de microescalas e microtexturas de peneiras.

Posição (mm)		Posição (mm)		Posição (mm)		Posição (mm)		Posição (mm)	
Nominal	Medida	Nominal	Medida	Nominal	Medida	Nominal	Medida	Nominal	Medida
0,00	0,0000	0,22	0,2192	0,42	0,4191	0,64	0,6391	0,84	0,8393
0,02	0,0192	0,24	0,2392	0,44	0,4392	0,66	0,6592	0,86	0,8593
0,04	0,0392	0,25	0,2494	0,46	0,4592	0,68	0,6792	0,88	0,8793
0,06	0,0591	0,26	0,2593	0,48	0,4792	0,70	0,6993	0,90	0,8994
0,08	0,0790	0,28	0,2793	0,50	0,4993	0,72	0,7193	0,92	0,9193
0,10	0,0991	0,30	0,2995	0,52	0,5192	0,74	0,7393	0,94	0,9392
0,12	0,1190	0,32	0,3192	0,54	0,5391	0,75	0,7494	0,96	0,9591
0,14	0,1390	0,34	0,3394	0,56	0,5592	0,76	0,7594	0,98	0,9792
0,16	0,1589	0,36	0,3592	0,58	0,5792	0,78	0,7794	1,00	0,9994
0,18	0,1790	0,38	0,3792	0,60	0,5993	0,80	0,7994	1,01	1,0092
0,20	0,1992	0,40	0,3992	0,62	0,6190	0,82	0,8193		
incerteza das medidas: ±0,0002 mm									

Tabela 4-1: Medidas-referência para microescala de 1 mm.

	Abertura da Malha (mm)			Diâmetro do Fio (mm)	
	Média	Tolerância	Máxima	Médio	Tolerância
Nominal	0,0440	±0,0031	0,0670	0,0320	±0,050
Medida	0,0440		0,0500	0,0350	
Incerteza das medidas: ±0,002 mm					

Tabela 4-2: Medidas-referência para peneira ABNT-325.

4.3 Software de Visão Computacional

A construção dos instrumentos metrológicos inteligentes, a partir da integração de algoritmos incorporados a bibliotecas de ferramentas comerciais de processamento de imagens disponíveis no IPT [3], foi considerada, de início, como uma alternativa bastante atraente. Esta, porém, acabou sendo descartada, após se verificar que os algoritmos implementados naquelas bibliotecas, de caráter muito geral, não atendiam a todas as necessidades e especificidades do problema em foco. Dessa forma, optou-se pela implementação integral da quase totalidade dos algoritmos necessários à automatização dos procedimentos de medição; a exceção ficou por conta de alguns algoritmos de análise numérica, diretamente copiados da biblioteca de software da referência (PRESS *et al*, 1990). Para a operação do sistema M-Vision 1000 foram aplicadas as funções de configuração de circuitos, digitalização e armazenamento de imagens disponíveis na biblioteca MV-1000.DLL (MUTECH, 1995).

Utilizando-se o compilador Visual C++ versão 2.0 (MICROSOFT, 1994) e a metodologia de desenvolvimento de aplicações Windows orientadas por mensagem (CALVERT, 1994), os diversos algoritmos de visão computacional implementados foram encapsulados na forma de instrumentos metrológicos virtuais cujos controles podem ser selecionados e ajustados pelo usuário através de dispositivos gráficos de uma interface Windows padrão. Além disso, foram também incorporadas ao sistema opções para a utilização independente e interativa dos algoritmos de visão computacional

disponíveis, bem como as funções essenciais ao seu funcionamento, tais como captura e gravação de imagens. Na figura 4.5 apresenta-se a barra de menus do sistema com as opções disponibilizadas ao usuário.

Figura 4.5: Barra de menus do Sistema de Metrologia Baseado em Visão Computacional.

Durante a implementação, inúmeros testes foram aplicados aos algoritmos, em geral utilizando-se imagens-referência de objetos artificiais como, por exemplo, as da figura 4.6, aplicadas à validação do algoritmo de estimação de freqüências espaciais.

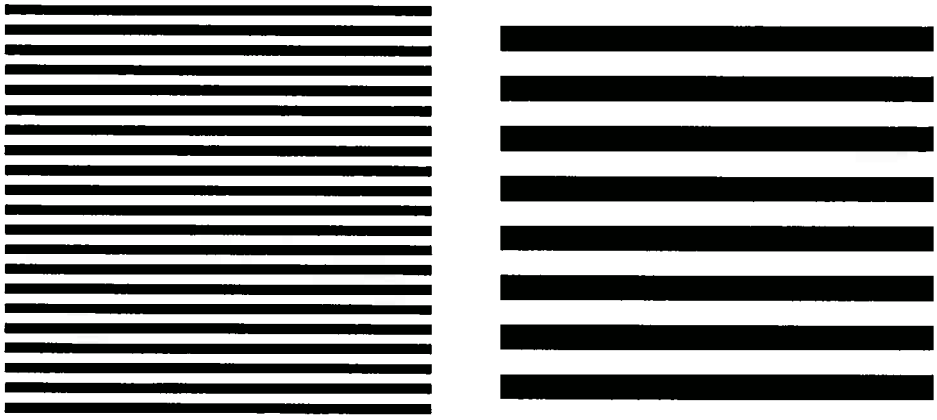

Figura 4.6: Imagens-teste para validação de algoritmos de visão computacional.

Apesar de não se poder afirmar, categoricamente, que não existam erros de codificação nos algoritmos implementados, os resultados dos testes, bem como os das subseqüentes aplicações a "imagens reais", mostraram-se bastante satisfatórios. Ademais, através da seleção da opção *Rastreamento*, apresentada no próximo tópico, registra-se a seqüência de imagens e arquivos de dados intermediários gerados pelos instrumentos computacionais, os quais poderão ser eventualmente utilizados em uma

[3] No caso, as ferramentas Visilog 5.0, da Noesis Vision Inc. e KS-400, da Kontron Systems.

bancada de testes do tipo "caixa-preta", para análise de conformidade dos algoritmos implementados com os correspondentes algoritmos referência (HOPP, 1993).

No Anexo I apresenta-se a listagem dos programas-fonte que implementam os diversos recursos e algoritmos de visão computacional disponibilizados pelo sistema, excluindo-se apenas as rotinas de construção da interface Windows e os algoritmos de cálculo numérico extraídos da referência (PRESS *et al*, 1990).

4.3.1 Funções para Operação no Modo Interativo

Operando como uma ferramenta de processamento de imagens convencional, o sistema oferece as seguintes opções ao usuário:

1. **Imagens:** Operações de gerenciamento de arquivos de imagens.

- ***Leitura***: Leitura de um arquivo do tipo "bitmap".
- ***Aquisição***: Captura de uma imagem através das funções de biblioteca do sistema M-Vision 1000 (MV-1000.DLL), configuradas para a recepção e digitalização de sinais de vídeo composto gerados pela câmera Optronics LE-470.
- ***Gravação***: Gravação de uma imagem como arquivo do tipo "bitmap".
- ***Seleção***: Apresentação de uma imagem selecionada.
- ***Eliminação***: Destruição de uma imagem selecionada.
- ***Rastreamento***: Gravação da seqüência de imagens e arquivos de dados intermediários gerados por um instrumento metrológico de visão computacional.

2. **Suavização:** Operações de filtragem espacial da imagem.

- ***Média***: Aplicação à imagem de um filtro-média de largura dada.
- ***Mediana***: Aplicação à imagem de um filtro-mediana de largura dada.
- ***Ruído***: Estimação do desvio-padrão do ruído da imagem.

3. **Limiarização:** Operações de limiarização e rotulação da imagem.

- ***Manual***: Limiarização da imagem utilizando um valor dado de limiar.
- ***Automática***: Limiarização da imagem utilizando um valor calculado de limiar.
- ***Rotulação***: Rotulação dos objetos de uma imagem binária.

4. **Aritmética:** Operações aritméticas e lógicas entre imagens.

- ***Or***: União de duas imagens.
- ***And***: Intersecção de duas imagens.
- ***Xor***: Diferenças de duas imagens.
- ***Not***: Inversão dos níveis de cinza de uma imagem.

5. **Bordas:** Detecção, limiarização, afinamento e localização das coordenadas das bordas de uma imagem, de acordo com uma das seguintes opções:

- ***Roberts***: Aplicação do detector de Roberts
- ***Sobel***: Aplicação do detector de Sobel
- ***Sub-Pixel 3x3***: Aplicação do detector de Jensen *et* Anastassiou
- ***Sub-Pixel 5x5***: Aplicação do detector de Lyvers *et* Mitchell.

6. **Morfologia:** Aplicação de operações de morfologia matemática a imagens binárias.

- ***Dilatação***: Dilatação da imagem com um elemento estruturante dado.
- ***Erosão***: Erosão da imagem com um elemento estruturante dado.
- ***Abertura***: Abertura da imagem com um elemento estruturante dado.
- ***Fechamento***: Fechamento da imagem com um elemento estruturante dado.
- **Bordas**: Detecção morfológica de bordas.
- ***Dilatação Condicional***: Dilatação condicional da $Imagem_1$ relativamente à $Imagem_2$.
- ***Destruição de Objetos Limítrofes***: Eliminação dos objetos incompletos da imagem.
- ***Esqueleto***: Determinação do esqueleto da imagem.

7. **Retas:** Localização e ajuste de retas ou de padrões repetitivos da imagem.

- ***Transformada de Hough***: Localização das retas via transformada de Hough.
- ***Ajuste MMQ***: Ajuste pelo método dos mínimos quadrados de retas cuja localização na imagem é conhecida de forma aproximada.
- ***Freqüências Espaciais***: Determinação das freqüências espaciais características de uma imagem contendo padrões repetitivos segundo uma das direções *x* ou *y*.

4.3.2 Instrumentos Metrológicos

Foram implementados sete Instrumentos Metrológicos de Visão Computacional, sendo cinco para medição de microescalas (IMVC-E_1 a IMVC-E_5) e dois para medição de microtexturas de peneiras (IMVC-P_1, IMVC-P_2).

À exceção dos instrumentos IMVC-E_1 e IMVC-P_2, que utilizam uma única imagem do objeto I^T e um arquivo de medidas-referência, os demais sempre podem operar de duas formas distintas: 1ª) estimando o fator de escala K (*mm/nº de pixels*) com auxílio de um par de imagens (I^T_1, I^T_2), deslocadas mecanicamente de uma distância D conhecida e, em seguida, utilizando esse fator para calcular as distâncias (em *mm*) dos diversos traços localizados naquelas imagens; 2ª) estimando o fator de escala K a partir do mapa gerado pelas posições dos traços de uma imagem-referência I^{REF} e suas medidas respectivas determinadas por processo metrológico convencional e registradas em um arquivo M^{REF} de medidas-referência.

Em quaisquer dos casos acima, contudo, a operação do instrumento consiste sempre em um processo de atribuição de valores aos seus parâmetros de controle, conforme será mostrado nos sub-ítens a seguir.

4.3.2.1 IMVC-E_1

Trata-se de um instrumento para determinação do espaçamento médio f_0 entre os traços da microescala, estando esta aproximadamente alinhada com um dos eixos x ou y do sistema de referência da imagem. Conforme mostrado no esquema operacional da figura 4.7, o único parâmetro requerido pelo instrumento é a direção de medição — horizontal (vertical) se os traços estiverem alinhados com o eixo y (x).

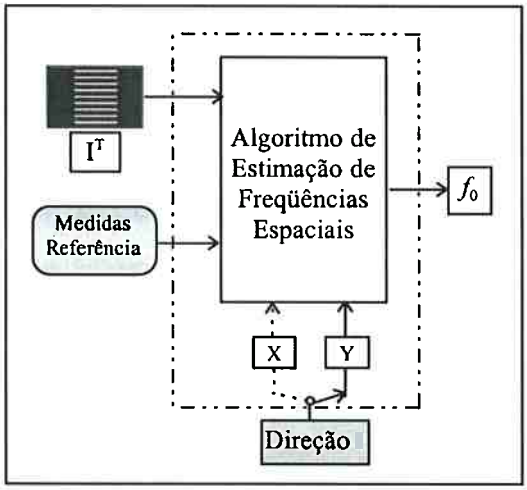

Figura 4.7: Esquema operacional do IMVC-E_1.

4.3.2.2 IMVC-E_2

Trata-se de um instrumento para determinação das posições dos traços da microescala, arbitrariamente orientada no plano-objeto, utilizando a transformada de Hough aplicada diretamente à sua imagem em tons de cinza. Conforme mostrado na caixa de diálogo da figura 4.8 e no esquema da figura 4.9, utiliza os parâmetros:

- *Ângulo mínimo* (θ_{min}): menor valor da coordenada angular do espaço de Hough;
- *Ângulo máximo* (θ_{max}): maior valor da coordenada angular do espaço de Hough;
- *Passo angular* ($d\theta$): incremento angular para a geração da transformada de Hough;
- *Comprimento* (L_{min}): menor comprimento esperado para os traços;

- *Espessura (E)*: meia-espessura estimada dos traços;
- *Limiar (L_h)*: valor do limiar da transformada de Hough (em % do valor máximo);
- *Espalhamento (s_ρ)*: espalhamento estimado dos picos no espaço de Hough;

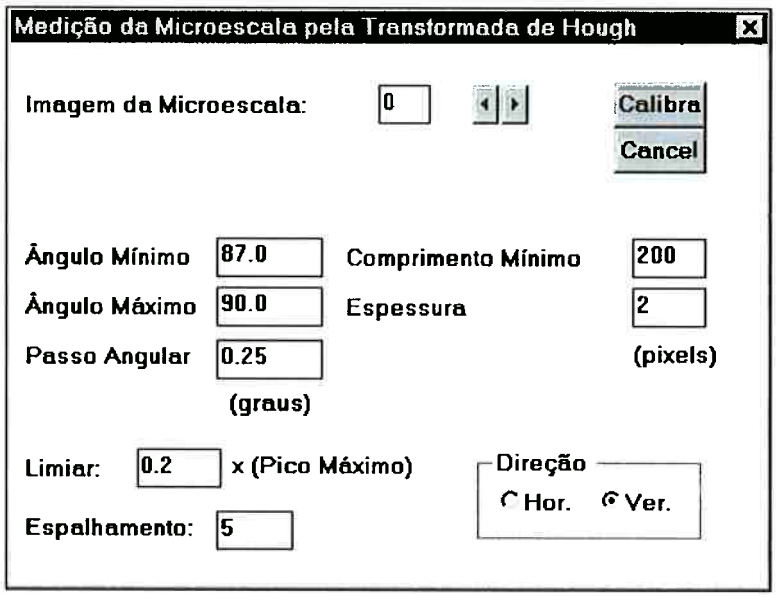

Figura 4.8: Caixa de diálogo para o IMVC-E_2.

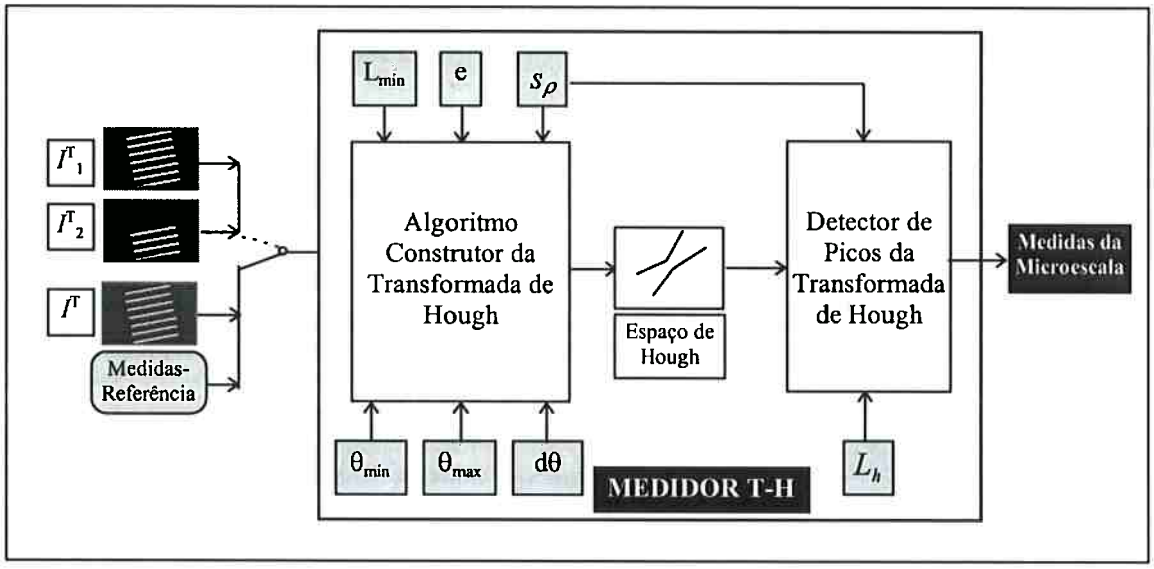

Figura 4.9: Esquema operacional do IMVC-E_2.

4.3.2.3 IMVC-E_3

Trata-se de um instrumento para determinação das posições dos traços da microescala através da aplicação de uma variante do método evolutivo (apresentado no

capítulo anterior), onde as medidas em escala nominal são obtidas através de limiarização e esqueletonização da imagem. Conforme ilustrado nas figuras 4.10 e 4.11, os parâmetros requeridos por este instrumento, são:

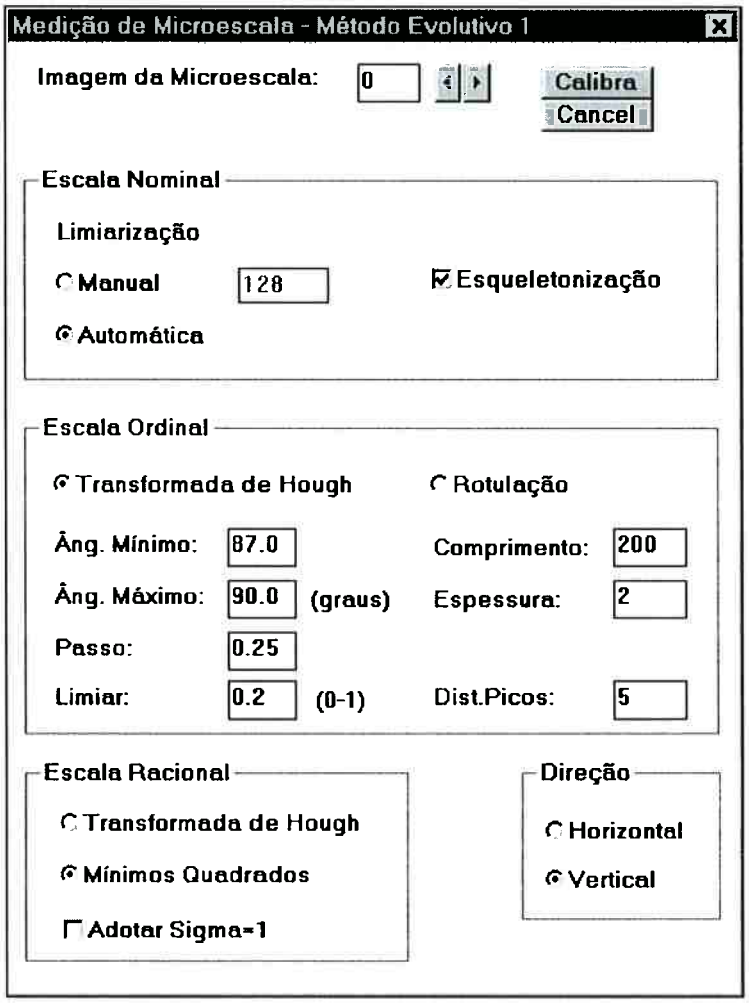

Figura 4.10: Caixa de diálogo para o IMVC-E_3.

A) da Escala Nominal:

- *Método de limiarização*: manual (valor arbitrado) ou automática (valor calculado);
- *Limiar*: valor arbitrado do limiar;
- *Esqueletonização*: se selecionado, a imagem binária é esqueletonizada.

B) da Escala Ordinal:

- *Método de ordenação*: transformada de Hough ou rotulação;

- *Demais parâmetros*: idênticos aos do IMVC-E_2 .

C) da Escala Racional:

- *Localização das retas*: transformada de Hough ou método dos mínimos quadrados;
- *Sigma Unitário*: se selecionado, considera que a distribuição dos pontos dos conjuntos ordenados de pixels das retas localizados na imagem têm mesma variança.

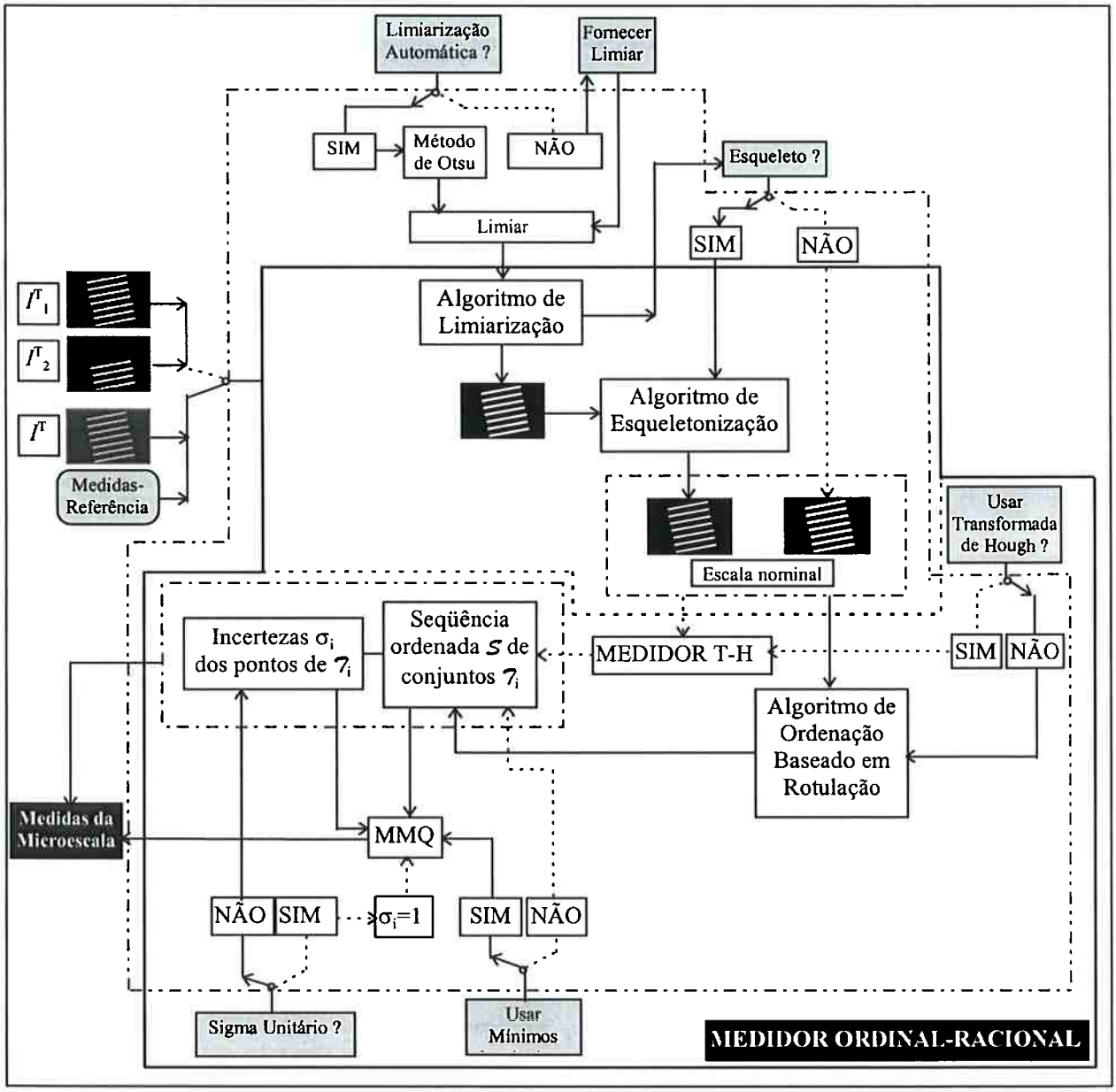

Figura 4.11: Esquema operacional do IMVC-E_3.

4.3.2.4 IMVC-E_4

Trata-se de um instrumento para determinação das posições dos traços da microescala através da aplicação de uma variante do método evolutivo, onde as medidas em escala nominal são obtidas via detecção e afinamento de bordas, conforme indicado nas figuras 4.12 e 4.13. Os parâmetros requeridos, são:

A) da Escala nominal:

- *Largura*: largura do filtro de suavização;
- *Reconstrução*: se selecionado, a imagem é reconstruída com dupla resolução;
- *Tipo de Borda*: detecção de bordas claro-escuras, bordas escuro-claras ou, então, da reta mediana entre as bordas limítrofes do traço;
- *Tipo de detector*: Roberts, Sobel, Jensen *et* Anastassiou ou Lyvers *et* Mitchell.

B,C) das Escalas ordinal e racional: idênticos aos do IMVC-E_3.

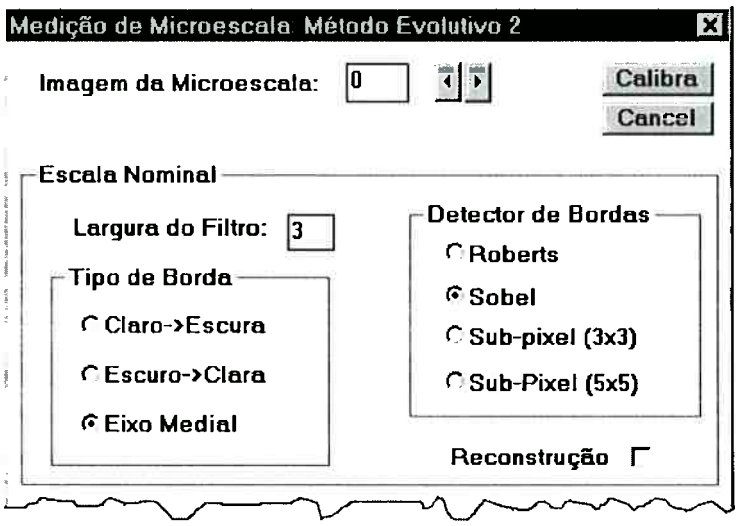

Figura 4.12: Caixa de diálogo para o IMVC-E_4.

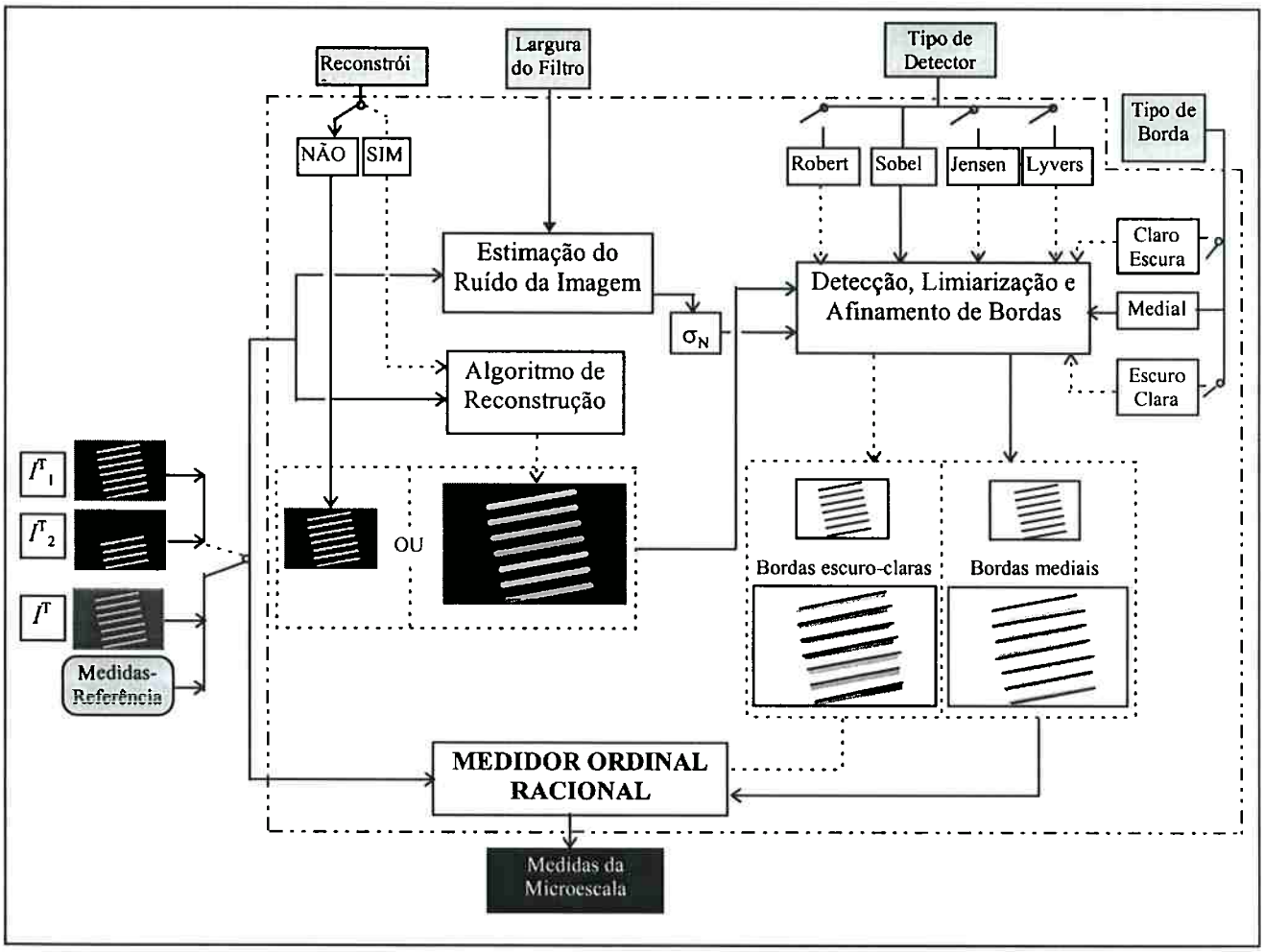

Figura 4.13: Esquema operacional do IMVC-E_4.

4.3.2.5 IMVC-E_5

Trata-se de um instrumento para determinação das posições dos traços da microescala através da aplicação de uma variante do método evolutivo em que as medições em escala nominal, inicialmente realizadas via limiarização automática, são submetidas a uma etapa posterior de refinamento, conforme se apresenta nas figuras 4.14 e 4.15. Os parâmetros requeridos, são:

A) da Escala nominal:

- *Tipo de Refinamento*: localização dos pontos de mínimo da imagem em tons de cinza ou das bordas utilizando um detector especial de bordas tipo *linha*.

B,C) das Escalas ordinal e racional: idênticos ao do IMVC-E_3.

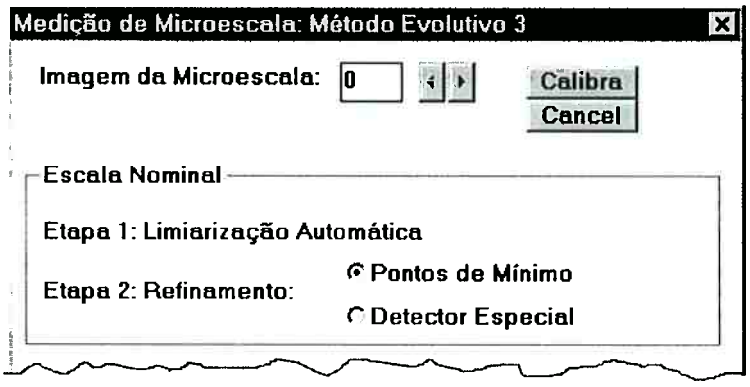

Figura 4.14: Caixa de diálogo para o IMVC-E_5.

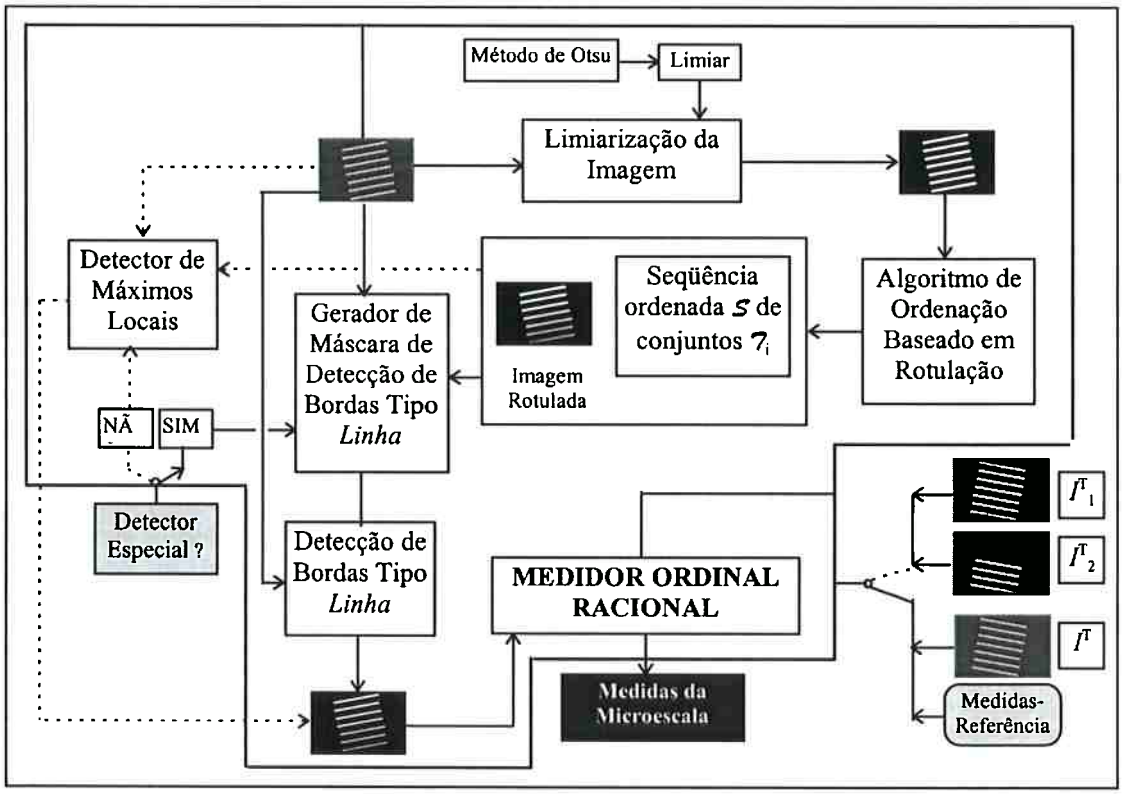

Figura 4.15: Esquema operacional do IMVC-E_5.

4.3.2.6 IMVC-P_1

Conforme se apresenta no esquema operacional da figura na figura 4.16, trata-se de instrumento para estimação da abertura média da trama e do diâmetro médio dos fios

de microtexturas de peneiras, utilizando variante do método evolutivo baseado nas ferramentas de medição abaixo.

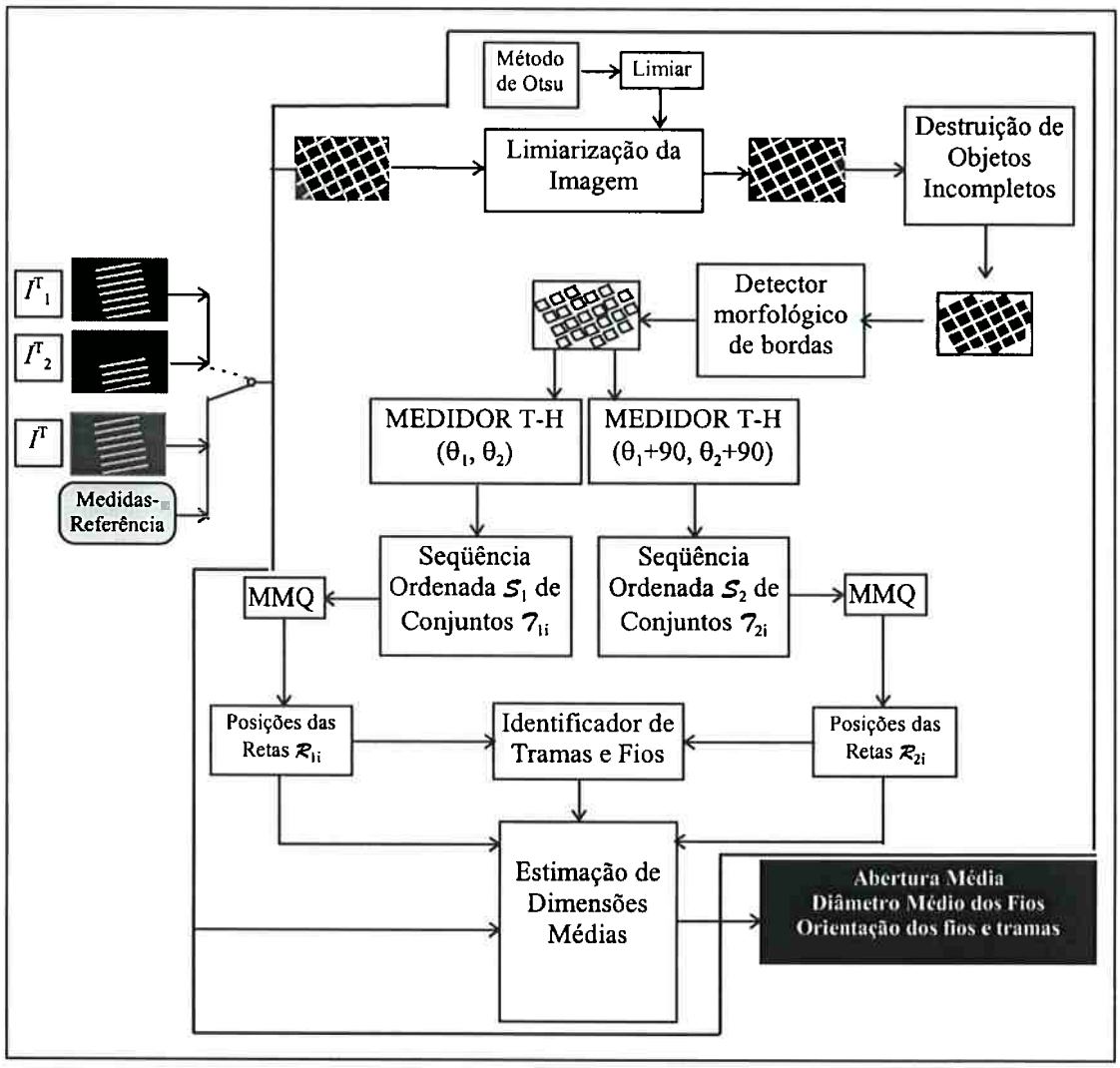

Figura 4.16: Esquema operacional do IMVC-P$_1$.

A) da Escala Nominal: seleção de pixels da imagem via limiarização, eliminação de objetos incompletos e aplicação de detector morfológico de bordas [4];

B) da Escala ordinal: Criação, através da transformada de Hough, de duas seqüências ordenadas S_1 e S_2 de conjuntos, cada qual agregando pixels de uma das duas direções ortogonais características da trama;

[4] Para a imagem I, definido como: Esqueleto{I⊕Circ(R=1)-I⊖Circ(R=1)}.

C) da Escala racional: Ajuste de retas através dos pontos de cada um dos conjuntos, via método dos mínimos quadrados.

4.3.2.7 IMVC-P_2

Conforme apresentado no esquema operacional da figura 4.17, trata-se de um instrumento para verificação das tolerâncias das aberturas das tramas e diâmetros dos fios da microtextura de peneiras, utilizando o método morfológico apresentado no capítulo 3. Para operar, necessita de informações geradas pelo IMVC-P_1: fatores de escala K_x, K_y e orientações θ_1, θ_2 das retas da imagem.

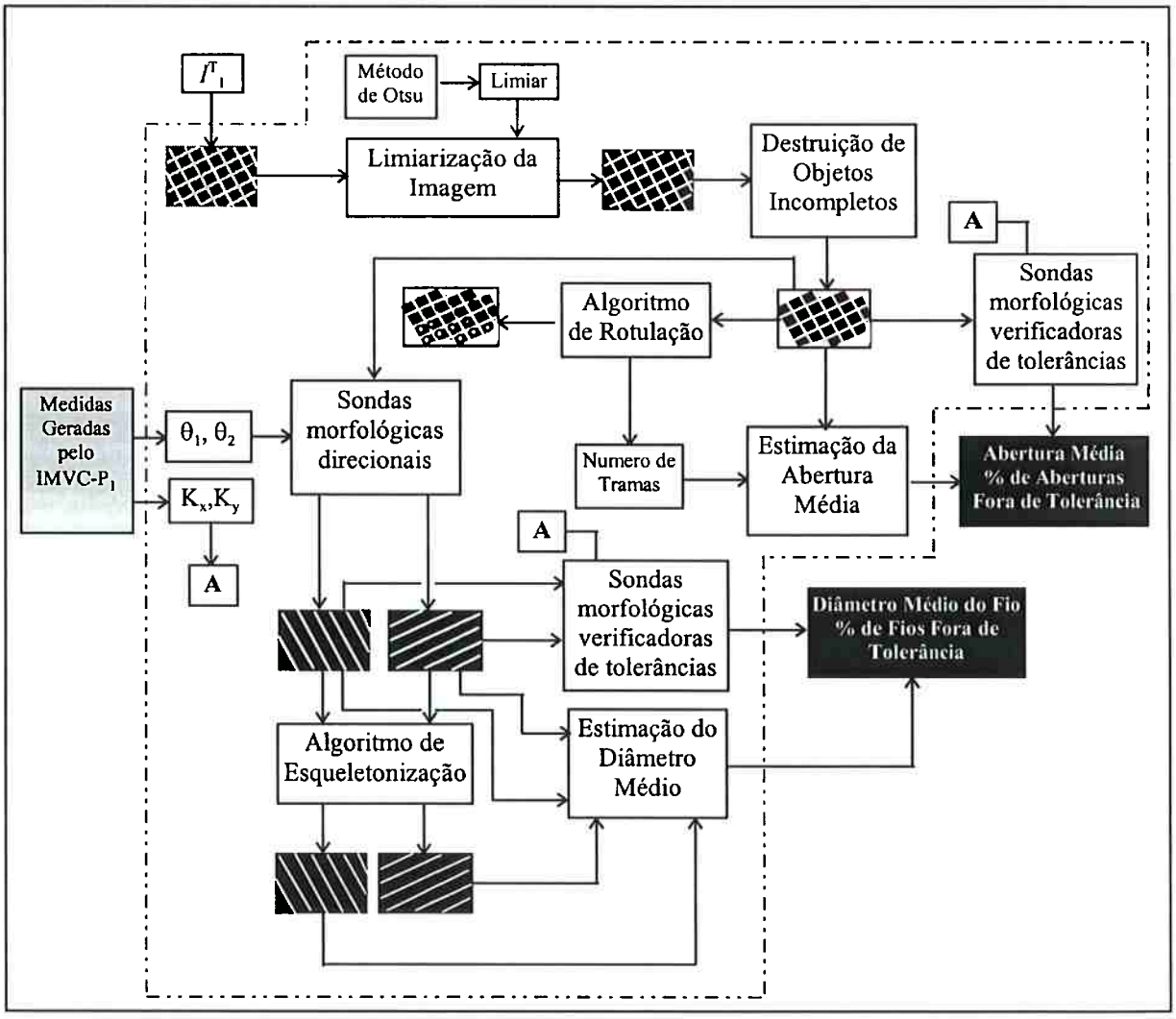

Figura 4.17: Esquema operacional do IMVC-P_2

5 - MEDIÇÕES CRIATIVAS E ROTINEIRAS

Antes de se aplicar os instrumentos metrológicos de visão computacional citados no capítulo anterior a ensaios rotineiros de calibração de microescalas e microtexturas de peneiras, é necessário realizar experimentos que, explicitando as relações entre exatidão (E), repetibilidade (R) e configuração do instrumento (CI), permitem concluir que, para aumentar E e R devem-se adotar providências sobre CI, como, por exemplo, definição de faixas de valores para os parâmetros p_j, j=1,...,m das ferramentas F_i, i=1,...,n ou substituição de uma ferramenta F_i por outra F_j. Em outras palavras, os referidos experimentos inserem-se no escopo das *medições criativas* (GNIOTEK, 1997), constituindo-se métodos de inferência das estruturas $F_e(p_1, p_2, ..., p_m)$ e $F_p(q_1, q_2, ...,q_n)$ dos modelos de medição de microescalas e microtexturas de peneiras, observados através dos seus correspondentes instrumentos metrológicos de visão computacional operando em condições definidas pelos vetores de parâmetros **P**, **Q**.

É importante mencionar que são componentes desses vetores não apenas os parâmetros apresentados no capítulo anterior, requeridos como dados de entrada dos IMVC's, mas também quatro outros fatores, determinados pelo operador no momento da aquisição da imagem: constante de aumento geométrico, potência da fonte de iluminação do microscópio, distância focal e inclinação do objeto no plano-imagem.

Nos tópicos que seguem discute-se, inicialmente, o efeito desses quatro fatores sobre a exatidão e repetibilidade das medidas e, imediatamente após, são apresentados os demais experimentos de avaliação de desempenho dos instrumentos implementados. Cabe destacar que as medidas em mm, reportadas nesses experimentos, foram obtidas através de calibração baseada em medidas-referência, conforme descrito no capítulo 3.

5.1 Constante de Aumento Geométrico

Do ponto de vista logístico, dever-se-ia adotar um fator de aumento geométrico para o ensaio de calibração de microescalas tal que todos os seus traços pudessem ser observados em uma única imagem. Para a instrumentação disponível, isso implicaria em se utilizar a objetiva do microscópio de menor aumento (3,2x). Todavia, conforme se pode observar na figura 5.1-a, imagens da microescala geradas com esse aumento têm resolução inadequada para a aplicação dos algoritmos de medição. Ademais, a tentativa de se aprimorar tais imagens mostrou-se infrutífera, como se pode observar na figura 5.1-b, onde a baixa definição das bordas persiste após reconstrução da imagem 5.1-a através do algoritmo de NALWA (1987).

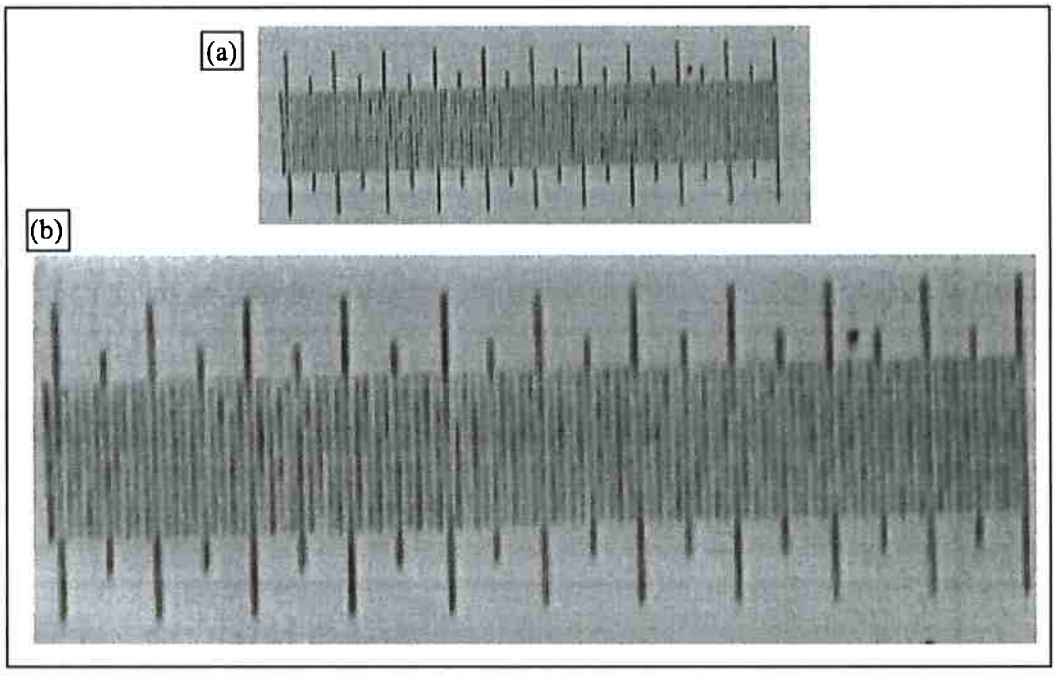

Figura 5.1: (a) Imagem de microescala de 1mm ampliada 96x; (b) Mesma imagem após reconstrução.

Utilizando-se uma objetiva com aumento de 10x, obtêm-se imagens da microescala de boa qualidade (vide figura 3.9) e, além disso, o ensaio de calibração pode ser realizado em duas etapas apenas: capturando-se duas imagens apresentando os traços 1-51 e 51-100, caso o alinhamento seja com a horizontal, ou os traços 1-69 e 32-

100, caso estes estejam alinhados com a vertical. Em quaisquer desses casos, contudo, é suficiente monitorar as posições do primeiro e últimos traços da microescala e utilizar o traço central (51) como referência para as medidas (figura 5.2).

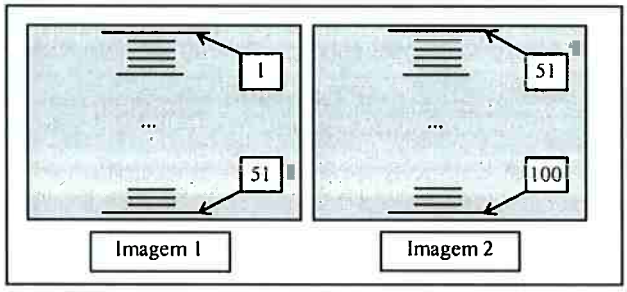

Figura 5.2: Esquema de captura de imagens para o ensaio de calibração de microescalas de 1mm.

Embora a utilização de objetivas com aumentos superiores (20x e 50x) produza imagens com maior resolução, a execução do ensaio torna-se muito mais complexa para o operador, pois se faz necessário capturar mais imagens e garantir a presença de um traço-referência nas posições inicial e final de cada par da seqüência de imagens. Por esse motivo, a objetiva de 10x foi adotada como padrão para os ensaios de calibração de microescalas.

Para o ensaio de calibração de microtexturas de peneiras, adotou-se a objetiva com aumento de 3,2x, pois as imagens capturadas mostraram-se de boa qualidade (vide figura 3.32) e apresentando amplificação (96x) comparável à do ensaio-referência baseado em projetor de perfis (100x). Ademais, a utilização de objetivas de maior aumento diminuiria o número de aberturas observadas, obrigando o operador a capturar mais imagens para validar estatisticamente as medidas.

5.2 Potência da Fonte de Iluminação

Para investigar a influência da intensidade luminosa média da imagem sobre as medidas geradas pelos IMVC's, capturaram-se três imagens da microescala, cada qual

correspondendo a um dos níveis *baixo*, *médio* e *alto* de potência da fonte de luz do microscópio, selecionados através do potenciômetro desse instrumento (ver figura 5.3). Por ser a escala nominal aquela que, eventualmente, sofreria influência direta das condições de iluminação, aplicaram-se os instrumentos IMVC-E_3 e IMVC-E_4, pois esses produzem medidas naquela escala através de métodos bastante distintos, baseados em limiarização e detecção de bordas. Em ambos os casos, estabeleceu-se uma única parametrização para os instrumentos, pois não se pretendia otimizar o seu desempenho, e sim avaliar a sua sensibilidade a alterações dos níveis de iluminação. Às 3 imagens referidas aplicaram-se, portanto, as operações:

1. IMVC-E_3 {EscNom:[Limiarização=Automática; Esqueleto=Não];EscOrd:[P_O];EscRac[P_R]} e

2. IMVC-E_4 {EscNom:[Filtro=Média(3);Reconstrói=Não;Detector=Sobel;EixoMedial=Sim];EscOrd: [P_O];EscRac:[P_R]}, onde P_O e P_R são os parâmetros selecionados para as medições em escala ordinal e racional: P_O[Método=Rotulação] e P_R[Método=MMQ; SigmaUnitário=Não].

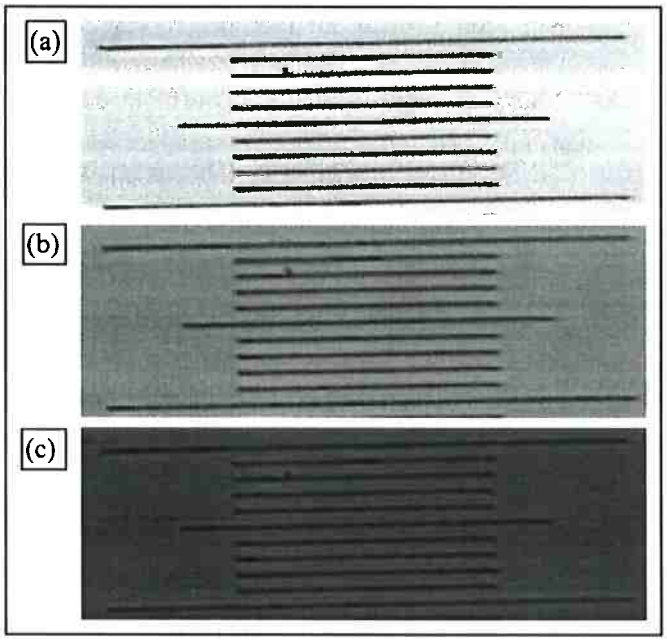

Figura 5.3: Variação da potência da fonte de iluminação: (a) alta; (b) média; (c) baixa.

Nas tabelas 5-1 e 5-2 apresentam-se, respectivamente, as posições d_i dos traços da microescala obtidas a partir da aplicação das operações 1 e 2 acima às três imagens iluminadas com os níveis de potência *alta*, *média* e *baixa*. Nas mesmas tabelas também são mostrados os desvios D_i das variáveis d_i das três amostras em relação aos seus valores médios dm_i ($D_i=|d_i-dm_i|$).

Traço	Potência Alta	Potência Média	Potência Baixa		Potência Alta	Potência Média	Potência Baixa
	Posições dos Traços (mm)			Média		Desvios	
	d	d	d	dm	D=|d-dm|	D=|d-dm|	D=|d-dm|
1	-0,0004	-0,0004	-0,0004	-0,000400	0,000000	0,000000	0,000000
2	0,0093	0,0094	0,0093	0,009333	0,000033	0,000067	0,000033
3	0,0191	0,0192	0,0192	0,019167	0,000067	0,000033	0,000033
4	0,0291	0,0292	0,0291	0,029133	0,000033	0,000067	0,000033
5	0,0390	0,0390	0,0389	0,038967	0,000033	0,000033	0,000067
6	0,0490	0,0490	0,0489	0,048967	0,000033	0,000033	0,000067
7	0,0591	0,0592	0,0592	0,059167	0,000067	0,000033	0,000033
8	0,0691	0,0690	0,0691	0,069067	0,000033	0,000067	0,000033
9	0,0791	0,0790	0,0790	0,079033	0,000067	0,000033	0,000033
10	0,0890	0,0890	0,0889	0,088967	0,000033	0,000033	0,000067
11	0,0992	0,0992	0,0993	0,099233	0,000033	0,000033	0,000067
12	0,1092	0,1092	0,1092	0,109200	0,000000	0,000000	0,000000
13	0,1191	0,1191	0,1191	0,119100	0,000000	0,000000	0,000000
14	0,1291	0,1291	0,1291	0,129100	0,000000	0,000000	0,000000
15	0,1392	0,1391	0,1393	0,139200	0,000000	0,000100	0,000100
16	0,1492	0,1493	0,1493	0,149267	0,000067	0,000033	0,000033
17	0,1591	0,1592	0,1592	0,159167	0,000067	0,000033	0,000033
18	0,1691	0,1691	0,1691	0,169100	0,000000	0,000000	0,000000
19	0,1793	0,1791	0,1795	0,179300	0,000000	0,000200	0,000200
20	0,1892	0,1892	0,1893	0,189233	0,000033	0,000033	0,000067
21	0,1993	0,1993	0,1993	0,199300	0,000000	0,000000	0,000000
22	0,2092	0,2093	0,2093	0,209267	0,000067	0,000033	0,000033
23	0,2192	0,2192	0,2192	0,219200	0,000000	0,000000	0,000000
24	0,2294	0,2294	0,2294	0,229400	0,000000	0,000000	0,000000
25	0,2393	0,2393	0,2393	0,239300	0,000000	0,000000	0,000000
26	0,2493	0,2493	0,2493	0,249300	0,000000	0,000000	0,000000
27	0,2593	0,2592	0,2592	0,259233	0,000067	0,000033	0,000033
28	0,2692	0,2691	0,2691	0,269133	0,000067	0,000033	0,000033
29	0,2793	0,2793	0,2794	0,279333	0,000033	0,000033	0,000067
30	0,2893	0,2893	0,2893	0,289300	0,000000	0,000000	0,000000
31	0,2994	0,2994	0,2993	0,299367	0,000033	0,000033	0,000067
32	0,3093	0,3092	0,3092	0,309233	0,000067	0,000033	0,000033
33	0,3192	0,3193	0,3193	0,319267	0,000067	0,000033	0,000033
34	0,3293	0,3292	0,3293	0,329267	0,000033	0,000067	0,000033
35	0,3392	0,3392	0,3392	0,339200	0,000000	0,000000	0,000000
36	0,3491	0,3492	0,3491	0,349133	0,000033	0,000067	0,000033
37	0,3592	0,3590	0,3590	0,359067	0,000133	0,000067	0,000067
38	0,3691	0,3690	0,3692	0,369100	0,000000	0,000100	0,000100
39	0,3791	0,3791	0,3791	0,379100	0,000000	0,000000	0,000000
40	0,3891	0,3892	0,3891	0,389133	0,000033	0,000067	0,000033
41	0,3994	0,3992	0,3991	0,399233	0,000167	0,000033	0,000133
42	0,4092	0,4093	0,4094	0,409300	0,000100	0,000000	0,000100
43	0,4192	0,4192	0,4192	0,419200	0,000000	0,000000	0,000000
44	0,4291	0,4292	0,4292	0,429167	0,000067	0,000033	0,000033
45	0,4390	0,4391	0,4390	0,439033	0,000033	0,000067	0,000033
46	0,4492	0,4492	0,4490	0,449133	0,000067	0,000067	0,000133
47	0,4591	0,4593	0,4594	0,459267	0,000167	0,000033	0,000133
48	0,4691	0,4692	0,4692	0,469167	0,000067	0,000033	0,000033

Tabela 5-1: Posições e desvios dos traços obtidos via IMVC-E_3 para diferentes potências de iluminação.

Traço	Potência Alta	Potência Média	Potência Baixa		Potência Alta	Potência Média	Potência Baixa
	Posições dos Traços (mm)			Média	Desvios		
	d	d	d	dm	D=|d-dm|	D=|d-dm|	D=|d-dm|
1	-0,0003	-0,0004	-0,0003	-0,0003	0,000033	-0,000067	0,000033
2	0,0093	0,0093	0,0093	0,0093	0,000000	0,000000	0,000000
3	0,0191	0,0191	0,0191	0,0191	0,000000	0,000000	0,000000
4	0,0291	0,0291	0,0291	0,0291	0,000000	0,000000	0,000000
5	0,0390	0,0390	0,0390	0,0390	0,000000	0,000000	0,000000
6	0,0490	0,0490	0,0490	0,0490	0,000000	0,000000	0,000000
7	0,0591	0,0591	0,0591	0,0591	0,000000	0,000000	0,000000
8	0,0691	0,0690	0,0690	0,0690	0,000067	0,000033	0,000033
9	0,0791	0,0791	0,0790	0,0791	0,000033	0,000033	0,000067
10	0,0891	0,0891	0,0891	0,0891	0,000000	0,000000	0,000000
11	0,0991	0,0991	0,0992	0,0991	0,000033	0,000033	0,000067
12	0,1091	0,1091	0,1091	0,1091	0,000000	0,000000	0,000000
13	0,1191	0,1191	0,1191	0,1191	0,000000	0,000000	0,000000
14	0,1292	0,1292	0,1291	0,1292	0,000033	0,000033	0,000067
15	0,1391	0,1392	0,1392	0,1392	0,000067	0,000033	0,000033
16	0,1492	0,1492	0,1492	0,1492	0,000000	0,000000	0,000000
17	0,1592	0,1592	0,1592	0,1592	0,000000	0,000000	0,000000
18	0,1692	0,1692	0,1692	0,1692	0,000000	0,000000	0,000000
19	0,1793	0,1792	0,1792	0,1792	0,000067	0,000033	0,000033
20	0,1892	0,1892	0,1892	0,1892	0,000000	0,000000	0,000000
21	0,1994	0,1993	0,1993	0,1993	0,000067	0,000033	0,000033
22	0,2093	0,2093	0,2094	0,2093	0,000033	0,000033	0,000067
23	0,2192	0,2193	0,2193	0,2193	0,000067	0,000033	0,000033
24	0,2293	0,2293	0,2293	0,2293	0,000000	0,000000	0,000000
25	0,2392	0,2392	0,2392	0,2392	0,000000	0,000000	0,000000
26	0,2493	0,2493	0,2493	0,2493	0,000000	0,000000	0,000000
27	0,2593	0,2593	0,2594	0,2593	0,000033	0,000033	0,000067
28	0,2692	0,2693	0,2692	0,2692	0,000033	0,000067	0,000033
29	0,2793	0,2793	0,2793	0,2793	0,000000	0,000000	0,000000
30	0,2892	0,2892	0,2893	0,2892	0,000033	0,000033	0,000067
31	0,2994	0,2994	0,2994	0,2994	0,000000	0,000000	0,000000
32	0,3093	0,3092	0,3092	0,3092	0,000067	0,000033	0,000033
33	0,3192	0,3192	0,3192	0,3192	0,000000	0,000000	0,000000
34	0,3292	0,3292	0,3292	0,3292	0,000000	0,000000	0,000000
35	0,3392	0,3391	0,3392	0,3392	0,000033	0,000067	0,000033
36	0,3492	0,3492	0,3492	0,3492	0,000000	0,000000	0,000000
37	0,3592	0,3592	0,3592	0,3592	0,000000	0,000000	0,000000
38	0,3691	0,3691	0,3691	0,3691	0,000000	0,000000	0,000000
39	0,3791	0,3791	0,3791	0,3791	0,000000	0,000000	0,000000
40	0,3891	0,3891	0,3892	0,3891	0,000033	0,000033	0,000067
41	0,3993	0,3993	0,3993	0,3993	0,000000	0,000000	0,000000
42	0,4093	0,4092	0,4092	0,4092	0,000067	0,000033	0,000033
43	0,4192	0,4192	0,4192	0,4192	0,000000	0,000000	0,000000
44	0,4292	0,4292	0,4292	0,4292	0,000000	0,000000	0,000000
45	0,4391	0,4391	0,4391	0,4391	0,000000	0,000000	0,000000
46	0,4492	0,4492	0,4492	0,4492	0,000000	0,000000	0,000000
47	0,4592	0,4592	0,4591	0,4592	0,000033	0,000033	0,000067
48	0,4691	0,4691	0,4692	0,4691	0,000033	0,000033	0,000067

Tabela 5-2: Posições e desvios dos traços obtidos via IMVC-E_4 para diferentes potências de iluminação.

A identificação da natureza da perturbação $\hat{D}$ — puramente aleatória ou dependente da potência de iluminação selecionada — foi realizada através do método estatístico de análise de variança (JURAN; GRYNA, 1978), admitindo-se nível de significância de 95%. Conforme se apresentam nas tabelas 5-3 e 5-4, geradas pelo módulo de análise de dados da ferramenta Microsoft Excel 7.0 (MICROSOFT, 1995), em ambos os casos não se pode rejeitar a hipótese de igualdade das médias dos desvios

D_i; em outras palavras, não existem evidências a nível de 95% de significância, de que os diferentes níveis de potência de iluminação interfiram nos resultados das medidas.

ANOVA: fator único (potência de iluminação)

Hipótese H0: $\mu_{Alta} = \mu_{Media} = \mu_{Baixa}$

Grupo	*Contagem*	*Soma*	*Média*	*Variância*		
Alta	48	0,0019	3,9583E-05	1,7332E-09		
Média	48	0,0017	3,5417E-05	1,3431E-09		
Baixa	48	0,0021	4,3056E-05	1,9366E-09		
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Entre grupos	1,4043E-09	2	7,0216E-10	0,42022008	0,65772218	3,06029335
Dentro dos grupos	2,356E-07	141	1,6709E-09			
Total	2,3701E-07	143				
Conclusão:	$F < F_{crítico} \Rightarrow \mu_{Alta} = \mu_{Media} = \mu_{Baixa}$					

Tabela 5-3: Teste da influência da potência de iluminação sobre as medidas geradas pelo IMVC-E_3.

ANOVA: fator único (potência de iluminação)

Hipótese H0: $\mu_{Alta} = \mu_{Media} = \mu_{Baixa}$

Grupo	*Contagem*	*Soma*	*Média*	*Variância*		
Alta	48	0,00087	1,8056E-05	6,1269E-10		
Média	48	0,00073	1,5278E-05	4,2356E-10		
Baixa	48	0,00093	1,9444E-05	7,0134E-10		
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Entre grupos	4,321E-10	2	2,1605E-10	0,37301587	0,68933182	3,06029335
Dentro dos grupos	8,1667E-08	141	5,792E-10			
Total	8,2099E-08	143				
Conclusão:	$F < F_{crítico} \Rightarrow \mu_{Alta} = \mu_{Media} = \mu_{Baixa}$					

Tabela 5-4: Teste da influência da potência de iluminação sobre as medidas geradas pelo IMVC-E_4.

5.3 Distância Focal

Para avaliar a influência de variações na distância focal sobre a qualidade das medidas utilizou-se um grupo de 6 imagens da microescala (figura 5.4) em que, em relação à imagem adotada como referência, a distância focal variava de df=0,02 mm a df=0,10 mm, em passos de 0,02 mm. A essas imagens aplicou-se a operação IMVC-E_4 {EscNom:[Filtro=média(5);Reconstrói=não;Detector=Sobel;EixoMedial=sim];EscOrd:[P_O];EscRac:[P_R]} onde P_O e P_R têm o mesmo significado anterior, obtendo-se as posições d_i dos traços e seus desvios e_i em relação aos valores de referência (e_i=|d_i-d_{REF}|) (tabela 5-5).

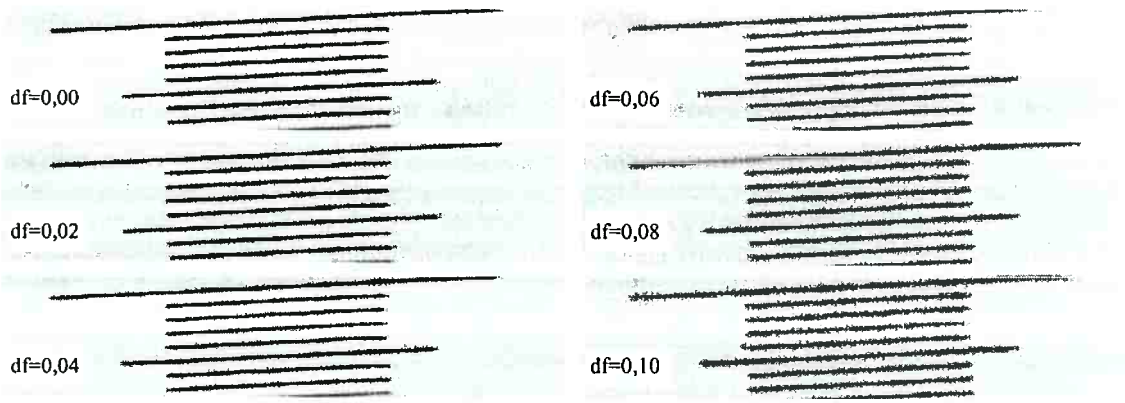

Figura 5.4: Imagens capturadas a diferentes distâncias focais.

		df=0,00		df=0,02		f=0,04		f=0,06		f=0,08		f=0,10	
T	d_{REF}	d	e	d	e	d	e	d	e	d	e	d	e
01	0,0000	-0,0003	0,0003	-0,0003	0,0003	-0,0003	0,0003	-0,0003	0,0003	-0,0004	0,0004	-0,0004	0,0004
02	0,0096	0,0092	0,0004	0,0093	0,0003	0,0093	0,0003	0,0093	0,0003	0,0093	0,0003	0,0093	0,0003
03	0,0192	0,0191	0,0001	0,0191	0,0001	0,0192	0,0000	0,0192	0,0000	0,0191	0,0001	0,0191	0,0001
04	0,0292	0,0290	0,0002	0,0290	0,0002	0,0290	0,0002	0,0290	0,0002	0,0289	0,0003	0,0290	0,0002
05	0,0392	0,0390	0,0002	0,0390	0,0002	0,0390	0,0002	0,0390	0,0002	0,0390	0,0002	0,0390	0,0002
06	0,0491	0,0490	0,0001	0,0490	0,0001	0,0490	0,0001	0,0490	0,0001	0,0490	0,0001	0,0490	0,0001
07	0,0591	0,0590	0,0001	0,0590	0,0001	0,0590	0,0001	0,0590	0,0001	0,0590	0,0001	0,0591	0,0000
08	0,0691	0,0690	0,0001	0,0690	0,0001	0,0690	0,0001	0,0691	0,0000	0,0690	0,0001	0,0690	0,0001
09	0,0790	0,0790	0,0000	0,0790	0,0000	0,0790	0,0000	0,0790	0,0000	0,0790	0,0000	0,0790	0,0000
10	0,0890	0,0890	0,0000	0,0890	0,0000	0,0890	0,0000	0,0890	0,0000	0,0891	0,0001	0,0891	0,0001
11	0,0991	0,0992	0,0001	0,0992	0,0001	0,0992	0,0001	0,0992	0,0001	0,0992	0,0001	0,0991	0,0000
12	0,1090	0,1091	0,0001	0,1091	0,0001	0,1090	0,0000	0,1090	0,0000	0,1091	0,0001	0,1091	0,0001
13	0,1190	0,1191	0,0001	0,1191	0,0001	0,1191	0,0001	0,1191	0,0001	0,1191	0,0001	0,1191	0,0001
14	0,1290	0,1291	0,0001	0,1291	0,0001	0,1291	0,0001	0,1291	0,0001	0,1291	0,0001	0,1291	0,0001
15	0,1390	0,1391	0,0001	0,1391	0,0001	0,1391	0,0001	0,1391	0,0001	0,1392	0,0002	0,1392	0,0002
16	0,1490	0,1492	0,0002	0,1492	0,0002	0,1492	0,0002	0,1492	0,0002	0,1493	0,0003	0,1493	0,0003
17	0,1589	0,1592	0,0003	0,1592	0,0003	0,1592	0,0003	0,1592	0,0003	0,1592	0,0003	0,1592	0,0003
18	0,1690	0,1692	0,0002	0,1691	0,0001	0,1691	0,0001	0,1692	0,0002	0,1693	0,0003	0,1693	0,0003
19	0,1790	0,1793	0,0003	0,1793	0,0003	0,1792	0,0002	0,1793	0,0003	0,1793	0,0003	0,1793	0,0003
20	0,1891	0,1892	0,0001	0,1892	0,0001	0,1892	0,0001	0,1892	0,0001	0,1892	0,0001	0,1892	0,0001
21	0,1992	0,1994	0,0002	0,1994	0,0002	0,1994	0,0002	0,1994	0,0002	0,1994	0,0002	0,1995	0,0003
22	0,2092	0,2093	0,0001	0,2093	0,0001	0,2093	0,0001	0,2093	0,0001	0,2093	0,0001	0,2093	0,0001
23	0,2192	0,2193	0,0001	0,2192	0,0000	0,2192	0,0000	0,2193	0,0001	0,2194	0,0002	0,2193	0,0001
24	0,2292	0,2294	0,0002	0,2294	0,0002	0,2294	0,0002	0,2294	0,0002	0,2294	0,0002	0,2294	0,0002
25	0,2392	0,2393	0,0001	0,2393	0,0001	0,2393	0,0001	0,2393	0,0001	0,2393	0,0001	0,2392	0,0000
26	0,2494	0,2493	0,0001	0,2494	0,0000	0,2494	0,0000	0,2493	0,0001	0,2494	0,0000	0,2494	0,0000
27	0,2593	0,2594	0,0001	0,2594	0,0001	0,2594	0,0001	0,2594	0,0001	0,2593	0,0000	0,2593	0,0000
28	0,2693	0,2693	0,0000	0,2693	0,0000	0,2693	0,0000	0,2692	0,0001	0,2693	0,0000	0,2693	0,0000
29	0,2793	0,2794	0,0001	0,2794	0,0001	0,2793	0,0000	0,2794	0,0001	0,2795	0,0002	0,2795	0,0002
30	0,2894	0,2894	0,0000	0,2894	0,0000	0,2893	0,0001	0,2893	0,0001	0,2892	0,0002	0,2892	0,0002
31	0,2995	0,2994	0,0001	0,2994	0,0001	0,2994	0,0001	0,2994	0,0001	0,2995	0,0000	0,2994	0,0001
32	0,3093	0,3094	0,0001	0,3093	0,0000	0,3094	0,0001	0,3093	0,0000	0,3094	0,0001	0,3093	0,0000
33	0,3192	0,3193	0,0001	0,3193	0,0001	0,3193	0,0001	0,3192	0,0000	0,3193	0,0001	0,3192	0,0000
34	0,3293	0,3293	0,0000	0,3293	0,0000	0,3293	0,0000	0,3293	0,0000	0,3294	0,0001	0,3294	0,0001
35	0,3394	0,3393	0,0001	0,3393	0,0001	0,3393	0,0001	0,3393	0,0001	0,3392	0,0002	0,3392	0,0002
36	0,3493	0,3492	0,0001	0,3492	0,0001	0,3492	0,0001	0,3492	0,0001	0,3493	0,0000	0,3493	0,0000
37	0,3592	0,3592	0,0000	0,3592	0,0000	0,3592	0,0000	0,3592	0,0000	0,3593	0,0001	0,3591	0,0001
38	0,3692	0,3692	0,0000	0,3692	0,0000	0,3692	0,0000	0,3691	0,0001	0,3691	0,0001	0,3689	0,0003
39	0,3792	0,3791	0,0001	0,3791	0,0001	0,3791	0,0001	0,3791	0,0001	0,3791	0,0001	0,3791	0,0001
40	0,3892	0,3892	0,0000	0,3892	0,0000	0,3892	0,0000	0,3892	0,0000	0,3891	0,0001	0,3890	0,0002
41	0,3992	0,3993	0,0001	0,3993	0,0001	0,3993	0,0001	0,3994	0,0002	0,3994	0,0002	0,3994	0,0002
42	0,4092	0,4092	0,0000	0,4092	0,0000	0,4092	0,0000	0,4092	0,0000	0,4092	0,0000	0,4090	0,0002
43	0,4191	0,4191	0,0000	0,4192	0,0001	0,4192	0,0001	0,4192	0,0001	0,4190	0,0001	0,4191	0,0000
44	0,4291	0,4291	0,0000	0,4291	0,0000	0,4291	0,0000	0,4291	0,0000	0,4291	0,0000	0,4290	0,0001
45	0,4392	0,4391	0,0001	0,4391	0,0001	0,4391	0,0001	0,4391	0,0001	0,4391	0,0001	0,4390	0,0002
46	0,4492	0,4492	0,0000	0,4491	0,0001	0,4491	0,0001	0,4491	0,0001	0,4491	0,0001	0,4492	0,0000
47	0,4592	0,4591	0,0001	0,4591	0,0001	0,4591	0,0001	0,4591	0,0001	0,4590	0,0002	0,4590	0,0002
48	0,4692	0,4691	0,0001	0,4691	0,0001	0,4691	0,0001	0,4691	0,0001	0,4691	0,0001	0,4689	0,0003
49	0,4792	0,4791	0,0001	0,4792	0,0000	0,4791	0,0001	0,4791	0,0001	0,4791	0,0001	0,4791	0,0001
50	0,4893	0,4891	0,0002	0,4890	0,0003	0,4891	0,0002	0,4891	0,0002	0,4890	0,0003	0,4895	0,0002
51	0,4993	0,4992	0,0001	0,4992	0,0001	0,4992	0,0001	0,4992	0,0001	0,4993	0,0000	0,4996	0,0003

Tabela 5.5: Desvios das posições dos traços em imagens com diferentes distâncias focais.

Conforme se observa nas tabelas 5-6 e 5-7, a análise de variância aplicada aos desvios e_i apresentados acima indica, com 95% de confiança, que variações da distância focal de até 0,06 mm não afetam o valor médio daqueles desvios, enquanto que para df= 0,10 mm já se observa nítida diferença relativamente aos valores médios dos quatro primeiros grupos de medidas. Considerando-se que as quatro imagens correspondendo a variações $df \leq 0{,}60$mm apresentam nitidez praticamente indistinguível à observação humana, mas que a partir de $df = 0{,}80$mm já se pode notar esmaecimento dos objetos da imagem, o resultado anterior é significativo, porque garante um intervalo seguro para o ajuste da distância focal do microscópio sem perda da qualidade das medidas.

ANOVA: fator único (variação da distância focal)

HIPÓTESE H0: $\mu_{0,00} = \mu_{0,20} = \mu_{0,40} = \mu_{0,60}$

Grupo	*Contagem*	*Soma*	*Média*	*Variância*		
df=0,00	51	0,0055	0,000107843	7,93725E-09		
df=0,20	51	0,0052	0,000101961	7,79608E-09		
df=0,40	51	0,005	9,80392E-05	6,59608E-09		
df=0,60	51	0,0055	0,000107843	7,13725E-09		
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Entre grupos	3,52941E-09	3	1,17647E-09	0,15970189	0,92332233	2,64975597
Dentro dos grupos	1,47333E-06	200	7,36667E-09			
Total	1,47686E-06	203				

CONCLUSÃO: $F < F_{crítico} \Rightarrow \mu_{0,00} = \mu_{0,20} = \mu_{0,40} = \mu_{0,60}$

Tabela 5-6: Teste 1 da influência da variação da distância focal sobre os desvios das medidas.

ANOVA: fator único (variação da distância focal)

HIPÓTESE H0: $\mu_{0,10} = \mu_{df \leq 0,60}$

Grupo	*Contagem*	*Soma*	*Média*	*Variância*		
df ≤ 0,60	51	0,0053	0,000103922	6,58431E-09		
df = 0,10	51	0,0073	0,000143137	1,2102E-08		
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Entre grupos	3,92157E-08	1	3,92157E-08	4,197271773	0,043108548	3,936150961
Dentro dos grupos	9,34314E-07	100	9,34314E-09			
Total	9,73529E-07	101				

CONCLUSÃO: $F > F_{crítico} \Rightarrow \mu_{0,10} \neq \mu_{df \leq 0,60}$

Tabela 5-7: Teste 2 da influência da variação da distância focal sobre os desvios das medidas.

5.4 Orientação do Objeto no Plano-Imagem

Para determinar se a particular orientação do objeto relativamente aos eixos do plano-imagem exercia influência sobre as medidas, capturaram-se sete grupos de cinco imagens orientadas segundo ângulos α com a vertical de 0,2°, 1,1°, 4,7°, 45,9°, 87,2°, 88,5° e 89,8° [1] deslocadas umas das outras de distâncias micrométricas arbitrárias, da ordem de 1 a 5 μm (ver figura 5.5). A essas imagens aplicou-se a operação IMVC-E_4 {EscNom:[Filtro=média(5);Reconstrói=não;Detector=Sobel;EixoMedial=sim];EscOrd:[P_O];EscRac:[P_R]} onde P_O e P_R têm o mesmo significado anterior.

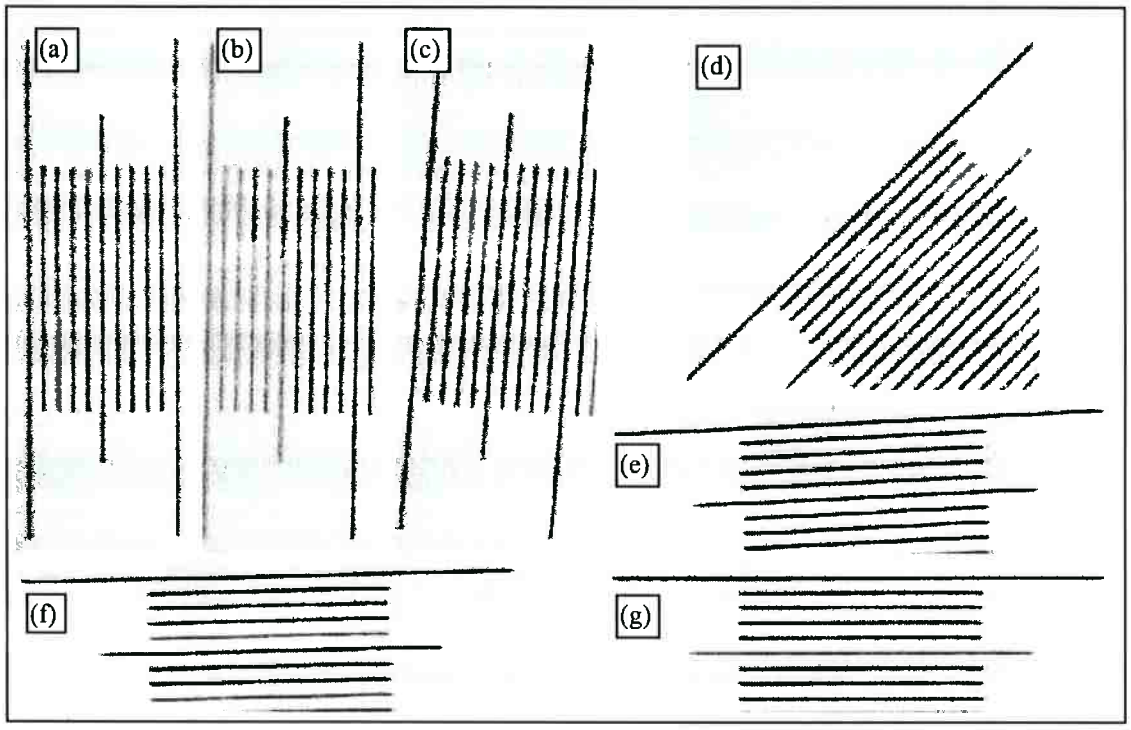

Figura 5.5: Traços orientados a: (a) 0,2 °; (b) 1,1 °; (c) 4,7 °; (d) 45,9 °; (e) 87,2 °; (f) 88,5 °; (g) 89,8 °.

Para cada um dos 7 grupos de 5 medidas d_{ij}, i=1,...,5, j=1,...n_T (onde n_T= número de traços da imagem *i*), determinadas pela operação acima, obtiveram-se as estimativas:

[1] Os valores de α foram determinados pelo próprio IMVC aplicado às imagens, após a microescala ter sido ligeiramente inclinada em relação às posições horizontal e vertical e também posicionada a aproximadamente 45°.

- valores médios $\bar{d}_j$ das posições dos traços $\mathcal{T}_j$ das imagens da microescala;

$$\bar{d}_j = \frac{1}{5}\sum_{i=1}^{5} d_{ij} \quad i = 1,\ldots,5 \tag{5-1}$$

- Desvios e_i das posições d_{ij} em relação às medidas-referência d_{REF} (ver tabela 4-1);

$$e_i = \left| d_{ij} - d_{REF} \right| \tag{5-2}$$

- Desvios-padrão s_j das posições d_j dos traços $\mathcal{T}_j$ das imagens da microescala;

$$s_j = \sqrt{\frac{1}{4}\sum_{i=1}^{5}\left(d_{ij} - \bar{d}_j\right)^2} \tag{5-3}$$

- Média σ_R dos desvios-padrão s_j;

$$\sigma_R = \frac{1}{n_T}\sum_{j=1}^{n_T} s_j \tag{5-4}$$

- Média quadrática σ_E dos desvios e_i.

$$\sigma_E = \sqrt{\frac{1}{n_T - 1}\left(e_i - \bar{e}\right)^2}, \quad onde \quad \bar{e} = \frac{1}{n_T}\sum_{j=1}^{n_T} \bar{e}_j \tag{5-5}$$

As medidas d_{ij} bem como as estimativas descritas através das equações 5-1 a 5-5 são apresentadas nas tabelas 5-8 a 5-14, enquanto que na figura 5.6 apresentam-se os gráficos de σ_E e σ_R em função do ângulo de orientação do objeto no plano-imagem.

Notando que σ_E é um indicador da discrepância entre o vetor de medidas médias $\bar{\mathbf{d}}$ e o vetor-referência $\mathbf{d_{REF}}$ e que σ_R representa o grau de dispersão das medidas $\hat{\mathbf{d}}$ repetidas em diferentes orientações e posições, conforme descrito acima, σ_E e σ_R serão consideradas, doravante, como as respectivas medidas de exatidão e repetibilidade [2] dos instrumentos metrológicos de visão computacional.

---

[2] Rigorosamente, σ_R mede uma propriedade intermediária entre repetibilidade e reprodutibilidade, uma vez que os experimentos foram realizados capturando-se, para cada posição angular arbitrariamente selecionada, imagens do mesmo objeto localizado em posições também arbitrárias.

					α= 0,2°				
T	d_1	d_2	d_3	d_4	d_5	d	s	d_{REF}	e |
1	-0,0004	-0,0006	-0,0006	-0,0005	-0,0006	-0,00054	0,000089	0,0000	0,00054
2	0,0091	0,0093	0,0090	0,0091	0,0092	0,00914	0,000114	0,0096	0,00046
3	0,0189	0,0189	0,0189	0,0192	0,0190	0,01898	0,000130	0,0192	0,00022
4	0,0291	0,0290	0,0290	0,0288	0,0291	0,02900	0,000122	0,0292	0,00020
5	0,0388	0,0391	0,0389	0,0390	0,0389	0,03894	0,000114	0,0392	0,00026
6	0,0489	0,0488	0,0490	0,0490	0,0488	0,04890	0,000100	0,0491	0,00020
7	0,0590	0,0590	0,0589	0,0589	0,0591	0,05898	0,000084	0,0591	0,00012
8	0,0690	0,0691	0,0689	0,0690	0,0689	0,06898	0,000084	0,0691	0,00012
9	0,0792	0,0791	0,0792	0,0790	0,0793	0,07916	0,000114	0,0790	0,00016
10	0,0889	0,0891	0,0890	0,0891	0,0889	0,08900	0,000100	0,0890	0,00000
11	0,0991	0,0989	0,0991	0,0990	0,0991	0,09904	0,000089	0,0991	0,00006
12	0,1091	0,1092	0,1089	0,1090	0,1092	0,10908	0,000130	0,1090	0,00008
13	0,1192	0,1192	0,1192	0,1192	0,1190	0,11916	0,000089	0,1190	0,00016
14	0,1293	0,1292	0,1294	0,1292	0,1292	0,12926	0,000089	0,1290	0,00026
15	0,1390	0,1391	0,1391	0,1393	0,1390	0,13910	0,000122	0,1390	0,00010
16	0,1493	0,1491	0,1493	0,1491	0,1493	0,14922	0,000110	0,1490	0,00022
17	0,1594	0,1594	0,1591	0,1592	0,1593	0,15928	0,000130	0,1589	0,00038
18	0,1691	0,1693	0,1694	0,1694	0,1691	0,16926	0,000152	0,1690	0,00026
19	0,1792	0,1793	0,1794	0,1793	0,1793	0,17930	0,000071	0,1790	0,00030
20	0,1892	0,1892	0,1892	0,1894	0,1892	0,18924	0,000089	0,1891	0,00014
21	0,1995	0,1994	0,1994	0,1993	0,1996	0,19944	0,000114	0,1992	0,00024
22	0,2094	0,2095	0,2093	0,2094	0,2095	0,20942	0,000084	0,2092	0,00022
23	0,2193	0,2193	0,2196	0,2195	0,2193	0,21940	0,000141	0,2192	0,00020
24	0,2293	0,2294	0,2295	0,2294	0,2295	0,22942	0,000084	0,2292	0,00022
25	0,2392	0,2394	0,2393	0,2394	0,2393	0,23932	0,000084	0,2392	0,00012
26	0,2496	0,2494	0,2495	0,2493	0,2495	0,24946	0,000114	0,2494	0,00006
27	0,2594	0,2595	0,2594	0,2595	0,2592	0,25940	0,000122	0,2593	0,00010
28	0,2693	0,2692	0,2694	0,2694	0,2693	0,26932	0,000084	0,2693	0,00002
29	0,2794	0,2795	0,2793	0,2794	0,2796	0,27944	0,000114	0,2793	0,00014
30	0,2894	0,2896	0,2893	0,2895	0,2895	0,28946	0,000114	0,2894	0,00006
31	0,2996	0,2995	0,2996	0,2995	0,2995	0,29954	0,000055	0,2995	0,00004
32	0,3092	0,3093	0,3094	0,3094	0,3092	0,30930	0,000100	0,3093	0,00000
33	0,3194	0,3191	0,3194	0,3192	0,3192	0,31926	0,000134	0,3192	0,00006
34	0,3295	0,3293	0,3292	0,3291	0,3295	0,32932	0,000179	0,3293	0,00002
35	0,3393	0,3394	0,3392	0,3394	0,3392	0,33930	0,000100	0,3394	0,00010
36	0,3493	0,3492	0,3494	0,3494	0,3493	0,34932	0,000084	0,3493	0,00002
37	0,3591	0,3594	0,3591	0,3592	0,3594	0,35924	0,000152	0,3592	0,00004
38	0,3691	0,3693	0,3692	0,3693	0,3692	0,36922	0,000084	0,3692	0,00002
39	0,3792	0,3792	0,3794	0,3791	0,3791	0,37920	0,000122	0,3792	0,00000
40	0,3891	0,3891	0,3893	0,3892	0,3890	0,38914	0,000114	0,3892	0,00006
41	0,3993	0,3992	0,3994	0,3993	0,3993	0,39930	0,000071	0,3992	0,00010
42	0,4093	0,4093	0,4090	0,4091	0,4094	0,40922	0,000164	0,4092	0,00002
43	0,4192	0,4193	0,4192	0,4193	0,4192	0,41924	0,000055	0,4191	0,00014
44	0,4291	0,4291	0,4293	0,4292	0,4292	0,42918	0,000084	0,4291	0,00008
45	0,4389	0,4391	0,4390	0,4390	0,4391	0,43902	0,000084	0,4392	0,00018
46	0,4491	0,4490	0,4491	0,4491	0,4491	0,44908	0,000045	0,4492	0,00012
47	0,4592	0,4591	0,4591	0,4590	0,4590	0,45908	0,000084	0,4592	0,00012
48	0,4691	0,4689	0,4691	0,4691	0,4688	0,46900	0,000141	0,4692	0,00020
49	0,4792	0,4790	0,4790	0,4791	0,4792	0,47910	0,000100	0,4792	0,00010
50	0,4890	0,4891	0,4888	0,4891	0,4892	0,48904	0,000152	0,4893	0,00026
51	0,4993	0,4991	0,4992	0,4991	0,4991	0,49916	0,000089	0,4993	0,00014
52	0,5090	0,5091	0,5092	0,5090	0,5090	0,50906	0,000089	0,5092	0,00014
53	0,5189	0,5189	0,5190	0,5190	0,5189	0,51894	0,000055	0,5192	0,00026
54	0,5292	0,5290	0,5291	0,5290	0,5292	0,52910	0,000100	0,5292	0,00010
55	0,5392	0,5393	0,5390	0,5391	0,5392	0,53916	0,000114	0,5391	0,00006
56	0,5492	0,5491	0,5492	0,5492	0,5491	0,54916	0,000055	0,5492	0,00004
57	0,5589	0,5590	0,5591	0,5591	0,5591	0,55904	0,000089	0,5592	0,00016
58	0,5688	0,5690	0,5690	0,5690	0,5689	0,56894	0,000089	0,5692	0,00026
59	0,5792	0,5789	0,5792	0,5789	0,5790	0,57904	0,000152	0,5792	0,00016
60	0,5892	0,5891	0,5890	0,5890	0,5890	0,58906	0,000089	0,5893	0,00024
61	0,5992	0,5991	0,5994	0,5992	0,5991	0,59920	0,000122	0,5993	0,00010
62	0,6091	0,6092	0,6092	0,6092	0,6093	0,60920	0,000071	0,6091	0,00010
63	0,6190	0,6192	0,6191	0,6192	0,6191	0,61912	0,000084	0,6190	0,00012
64	0,6293	0,6291	0,6294	0,6291	0,6292	0,62922	0,000130	0,6291	0,00012
65	0,6391	0,6391	0,6392	0,6391	0,6390	0,63910	0,000071	0,6391	0,00000
66	0,6492	0,6491	0,6493	0,6492	0,6492	0,64920	0,000071	0,6492	0,00000
67	0,6593	0,6593	0,6591	0,6593	0,6594	0,65928	0,000110	0,6592	0,00008
68	0,6692	0,6694	0,6692	0,6694	0,6693	0,66930	0,000100	0,6692	0,00010
69	0,6795	0,6794	0,6796	0,6793	0,6794	0,67944	0,000114	0,6792	0,00024
							σ_R		σ_E
							0,000103		0,000107

Tabela 5-8: Medidas e estimativas dos experimentos com imagens a 0,2 °com a vertical.

$\alpha = 1{,}1°$

T	d_1	d_2	d_3	d_4	d_5	d	s	d_{REF}	e
1	-0,0005	-0,0006	-0,0006	-0,0005	-0,0005	-0,00054	0,000055	0,0000	0,00054
2	0,0091	0,0092	0,0091	0,0091	0,0092	0,00914	0,000055	0,0096	0,00046
3	0,0190	0,0190	0,0190	0,0190	0,0189	0,01898	0,000045	0,0192	0,00022
4	0,0289	0,0290	0,0289	0,0289	0,0290	0,02894	0,000055	0,0292	0,00026
5	0,0389	0,0389	0,0389	0,0389	0,0389	0,03890	0,000000	0,0392	0,00030
6	0,0489	0,0489	0,0489	0,0489	0,0489	0,04890	0,000000	0,0491	0,00020
7	0,0590	0,0590	0,0590	0,0590	0,0590	0,05900	0,000000	0,0591	0,00010
8	0,0689	0,0690	0,0690	0,0690	0,0690	0,06898	0,000045	0,0691	0,00012
9	0,0790	0,0790	0,0791	0,0790	0,0790	0,07902	0,000045	0,0790	0,00002
10	0,0890	0,0891	0,0890	0,0890	0,0890	0,08902	0,000045	0,0890	0,00002
11	0,0991	0,0991	0,0991	0,0991	0,0991	0,09910	0,000000	0,0991	0,00000
12	0,1091	0,1091	0,1090	0,1091	0,1091	0,10908	0,000045	0,1090	0,00008
13	0,1191	0,1191	0,1191	0,1191	0,1191	0,11910	0,000000	0,1190	0,00010
14	0,1292	0,1292	0,1292	0,1292	0,1292	0,12920	0,000000	0,1290	0,00020
15	0,1392	0,1392	0,1392	0,1392	0,1392	0,13920	0,000000	0,1390	0,00020
16	0,1492	0,1492	0,1492	0,1493	0,1493	0,14924	0,000055	0,1490	0,00024
17	0,1592	0,1592	0,1593	0,1593	0,1593	0,15926	0,000055	0,1589	0,00036
18	0,1693	0,1693	0,1693	0,1693	0,1692	0,16928	0,000045	0,1690	0,00028
19	0,1793	0,1793	0,1793	0,1793	0,1793	0,17930	0,000000	0,1790	0,00030
20	0,1893	0,1893	0,1893	0,1893	0,1893	0,18930	0,000000	0,1891	0,00020
21	0,1994	0,1994	0,1994	0,1994	0,1994	0,19940	0,000000	0,1992	0,00020
22	0,2094	0,2094	0,2094	0,2094	0,2094	0,20940	0,000000	0,2092	0,00020
23	0,2194	0,2193	0,2194	0,2194	0,2194	0,21938	0,000045	0,2192	0,00018
24	0,2294	0,2294	0,2294	0,2294	0,2294	0,22940	0,000000	0,2292	0,00020
25	0,2393	0,2394	0,2393	0,2394	0,2393	0,23934	0,000055	0,2392	0,00014
26	0,2494	0,2494	0,2494	0,2494	0,2494	0,24940	0,000000	0,2494	0,00000
27	0,2594	0,2595	0,2595	0,2594	0,2594	0,25944	0,000055	0,2593	0,00014
28	0,2694	0,2694	0,2694	0,2694	0,2694	0,26940	0,000000	0,2693	0,00010
29	0,2795	0,2795	0,2795	0,2795	0,2795	0,27950	0,000000	0,2793	0,00020
30	0,2894	0,2894	0,2894	0,2894	0,2894	0,28940	0,000000	0,2894	0,00000
31	0,2995	0,2995	0,2995	0,2995	0,2995	0,29950	0,000000	0,2995	0,00000
32	0,3094	0,3094	0,3094	0,3094	0,3094	0,30940	0,000000	0,3093	0,00010
33	0,3193	0,3193	0,3193	0,3193	0,3193	0,31930	0,000000	0,3192	0,00010
34	0,3294	0,3293	0,3293	0,3293	0,3293	0,32932	0,000045	0,3293	0,00002
35	0,3393	0,3393	0,3393	0,3393	0,3393	0,33930	0,000000	0,3394	0,00010
36	0,3493	0,3493	0,3493	0,3493	0,3493	0,34930	0,000000	0,3493	0,00000
37	0,3592	0,3592	0,3592	0,3593	0,3593	0,35924	0,000055	0,3592	0,00004
38	0,3692	0,3692	0,3691	0,3692	0,3691	0,36916	0,000055	0,3692	0,00004
39	0,3792	0,3792	0,3792	0,3792	0,3792	0,37920	0,000000	0,3792	0,00000
40	0,3892	0,3892	0,3892	0,3892	0,3892	0,38920	0,000000	0,3892	0,00000
41	0,3993	0,3993	0,3993	0,3993	0,3993	0,39930	0,000000	0,3992	0,00010
42	0,4092	0,4092	0,4092	0,4092	0,4092	0,40920	0,000000	0,4092	0,00000
43	0,4191	0,4192	0,4192	0,4192	0,4191	0,41916	0,000055	0,4191	0,00006
44	0,4292	0,4292	0,4292	0,4292	0,4292	0,42920	0,000000	0,4291	0,00010
45	0,4390	0,4390	0,4390	0,4391	0,4391	0,43904	0,000055	0,4392	0,00016
46	0,4491	0,4491	0,4491	0,4491	0,4491	0,44910	0,000000	0,4492	0,00010
47	0,4591	0,4591	0,4591	0,4591	0,4591	0,45910	0,000000	0,4592	0,00010
48	0,4691	0,4691	0,4690	0,4691	0,4690	0,46906	0,000055	0,4692	0,00014
49	0,4791	0,4791	0,4791	0,4791	0,4791	0,47910	0,000000	0,4792	0,00010
50	0,4890	0,4890	0,4890	0,4890	0,4890	0,48900	0,000000	0,4893	0,00030
51	0,4991	0,4991	0,4991	0,4992	0,4991	0,49912	0,000045	0,4993	0,00018
52	0,5091	0,5091	0,5091	0,5090	0,5090	0,50906	0,000055	0,5092	0,00014
53	0,5190	0,5190	0,5190	0,5190	0,5190	0,51900	0,000000	0,5192	0,00020
54	0,5291	0,5291	0,5291	0,5291	0,5291	0,52910	0,000000	0,5292	0,00010
55	0,5391	0,5391	0,5391	0,5391	0,5391	0,53910	0,000000	0,5391	0,00000
56	0,5491	0,5491	0,5491	0,5491	0,5491	0,54910	0,000000	0,5492	0,00010
57	0,5590	0,5590	0,5590	0,5590	0,5590	0,55900	0,000000	0,5592	0,00020
58	0,5690	0,5690	0,5690	0,5690	0,5690	0,56900	0,000000	0,5692	0,00020
59	0,5791	0,5791	0,5791	0,5791	0,5791	0,57910	0,000000	0,5792	0,00010
60	0,5891	0,5891	0,5891	0,5891	0,5891	0,58910	0,000000	0,5893	0,00020
61	0,5992	0,5992	0,5992	0,5992	0,5992	0,59920	0,000000	0,5993	0,00010
62	0,6092	0,6092	0,6092	0,6092	0,6092	0,60920	0,000000	0,6091	0,00010
63	0,6191	0,6191	0,6191	0,6191	0,6191	0,61910	0,000000	0,6190	0,00010
64	0,6292	0,6292	0,6292	0,6292	0,6292	0,62920	0,000000	0,6291	0,00010
65	0,6392	0,6392	0,6391	0,6391	0,6391	0,63914	0,000055	0,6391	0,00004
66	0,6492	0,6492	0,6492	0,6492	0,6492	0,64920	0,000000	0,6492	0,00000
67	0,6592	0,6593	0,6593	0,6593	0,6593	0,65928	0,000045	0,6592	0,00008
68	0,6692	0,6692	0,6692	0,6693	0,6692	0,66922	0,000045	0,6692	0,00002
69	0,6794	0,6794	0,6794	0,6794	0,6794	0,67940	0,000000	0,6792	0,00020
							σ_R		σ_E
							0,000018		0,000109

Tabela 5-9: Medidas e estimativas dos experimentos com imagens a 1,1 °com a vertical.

α = 4,7°

T	d_1	d_2	d_3	d_4	d_5	d	s	d_{REF}	e
1	-0,0005	-0,0005	-0,0005	-0,0005	-0,0005	-0,00050	0,000000	0,0000	0,00050
2	0,0092	0,0092	0,0092	0,0092	0,0092	0,00920	0,000000	0,0096	0,00040
3	0,0190	0,0190	0,0190	0,0190	0,0190	0,01900	0,000000	0,0192	0,00020
4	0,0290	0,0290	0,0290	0,0289	0,0290	0,02898	0,000045	0,0292	0,00022
5	0,0389	0,0389	0,0389	0,0389	0,0389	0,03890	0,000000	0,0392	0,00030
6	0,0489	0,0489	0,0489	0,0490	0,0489	0,04892	0,000045	0,0491	0,00018
7	0,0590	0,0590	0,0590	0,0590	0,0590	0,05900	0,000000	0,0591	0,00010
8	0,0690	0,0690	0,0690	0,0690	0,0690	0,06900	0,000000	0,0691	0,00010
9	0,0791	0,0790	0,0791	0,0790	0,0791	0,07906	0,000055	0,0790	0,00006
10	0,0890	0,0890	0,0890	0,0890	0,0890	0,08900	0,000000	0,0890	0,00000
11	0,0991	0,0991	0,0991	0,0991	0,0991	0,09910	0,000000	0,0991	0,00000
12	0,1091	0,1091	0,1091	0,1091	0,1091	0,10910	0,000000	0,1090	0,00010
13	0,1191	0,1191	0,1191	0,1191	0,1191	0,11910	0,000000	0,1190	0,00010
14	0,1292	0,1292	0,1292	0,1292	0,1292	0,12920	0,000000	0,1290	0,00020
15	0,1392	0,1392	0,1392	0,1392	0,1392	0,13920	0,000000	0,1390	0,00020
16	0,1493	0,1492	0,1492	0,1492	0,1493	0,14924	0,000055	0,1490	0,00024
17	0,1592	0,1592	0,1592	0,1593	0,1592	0,15922	0,000045	0,1589	0,00032
18	0,1692	0,1692	0,1692	0,1693	0,1692	0,16922	0,000045	0,1690	0,00022
19	0,1793	0,1793	0,1793	0,1793	0,1793	0,17930	0,000000	0,1790	0,00030
20	0,1893	0,1893	0,1893	0,1893	0,1893	0,18930	0,000000	0,1891	0,00020
21	0,1994	0,1994	0,1994	0,1994	0,1994	0,19940	0,000000	0,1992	0,00020
22	0,2094	0,2094	0,2094	0,2093	0,2094	0,20938	0,000045	0,2092	0,00018
23	0,2193	0,2194	0,2193	0,2194	0,2193	0,21934	0,000055	0,2192	0,00014
24	0,2294	0,2294	0,2294	0,2294	0,2294	0,22940	0,000000	0,2292	0,00020
25	0,2393	0,2393	0,2393	0,2393	0,2393	0,23930	0,000000	0,2392	0,00010
26	0,2494	0,2494	0,2494	0,2494	0,2494	0,24940	0,000000	0,2494	0,00000
27	0,2595	0,2594	0,2594	0,2594	0,2594	0,25942	0,000045	0,2593	0,00012
28	0,2694	0,2694	0,2694	0,2694	0,2694	0,26940	0,000000	0,2693	0,00010
29	0,2795	0,2794	0,2795	0,2795	0,2794	0,27946	0,000055	0,2793	0,00016
30	0,2894	0,2894	0,2894	0,2894	0,2894	0,28940	0,000000	0,2894	0,00000
31	0,2995	0,2995	0,2995	0,2995	0,2995	0,29950	0,000000	0,2995	0,00000
32	0,3094	0,3094	0,3094	0,3094	0,3094	0,30940	0,000000	0,3093	0,00010
33	0,3193	0,3193	0,3193	0,3193	0,3193	0,31930	0,000000	0,3192	0,00010
34	0,3293	0,3293	0,3293	0,3293	0,3293	0,32930	0,000000	0,3293	0,00000
35	0,3393	0,3393	0,3393	0,3393	0,3393	0,33930	0,000000	0,3394	0,00010
36	0,3493	0,3493	0,3493	0,3493	0,3493	0,34930	0,000000	0,3493	0,00000
37	0,3592	0,3592	0,3592	0,3592	0,3592	0,35920	0,000000	0,3592	0,00000
38	0,3691	0,3691	0,3691	0,3691	0,3691	0,36910	0,000000	0,3692	0,00010
39	0,3791	0,3792	0,3792	0,3792	0,3792	0,37918	0,000045	0,3792	0,00002
40	0,3892	0,3892	0,3892	0,3892	0,3892	0,38920	0,000000	0,3892	0,00000
41	0,3993	0,3993	0,3993	0,3993	0,3993	0,39930	0,000000	0,3992	0,00010
42	0,4092	0,4092	0,4092	0,4092	0,4092	0,40920	0,000000	0,4092	0,00000
43	0,4192	0,4192	0,4192	0,4191	0,4191	0,41916	0,000055	0,4191	0,00006
44	0,4292	0,4292	0,4292	0,4292	0,4292	0,42920	0,000000	0,4291	0,00010
45	0,4390	0,4391	0,4390	0,4391	0,4390	0,43904	0,000055	0,4392	0,00016
46	0,4491	0,4491	0,4491	0,4491	0,4491	0,44910	0,000000	0,4492	0,00010
47	0,4591	0,4591	0,4591	0,4591	0,4591	0,45910	0,000000	0,4592	0,00010
48	0,4690	0,4690	0,4691	0,4691	0,4691	0,46906	0,000055	0,4692	0,00014
49	0,4791	0,4791	0,4791	0,4791	0,4791	0,47910	0,000000	0,4792	0,00010
50	0,4890	0,4890	0,4890	0,4890	0,4890	0,48900	0,000000	0,4893	0,00030
51	0,4991	0,4991	0,4992	0,4991	0,4992	0,49914	0,000055	0,4993	0,00016
52	0,5091	0,5091	0,5090	0,5091	0,5090	0,50906	0,000055	0,5092	0,00014
53	0,5190	0,5190	0,5190	0,5190	0,5190	0,51900	0,000000	0,5192	0,00020
54	0,5291	0,5291	0,5291	0,5291	0,5291	0,52910	0,000000	0,5292	0,00010
55	0,5391	0,5391	0,5391	0,5391	0,5391	0,53910	0,000000	0,5391	0,00000
56	0,5491	0,5491	0,5491	0,5491	0,5491	0,54910	0,000000	0,5492	0,00010
57	0,5590	0,5590	0,5591	0,5591	0,5591	0,55906	0,000055	0,5592	0,00014
58	0,5690	0,5690	0,5690	0,5690	0,5690	0,56900	0,000000	0,5692	0,00020
59	0,5791	0,5791	0,5791	0,5791	0,5791	0,57910	0,000000	0,5792	0,00010
60	0,5891	0,5891	0,5891	0,5891	0,5891	0,58910	0,000000	0,5893	0,00020
61	0,5992	0,5993	0,5993	0,5992	0,5993	0,59926	0,000055	0,5993	0,00004
62	0,6092	0,6092	0,6092	0,6092	0,6092	0,60920	0,000000	0,6091	0,00010
63	0,6191	0,6191	0,6191	0,6191	0,6192	0,61912	0,000045	0,6190	0,00012
64	0,6292	0,6292	0,6292	0,6292	0,6292	0,62920	0,000000	0,6291	0,00010
65	0,6392	0,6392	0,6392	0,6392	0,6392	0,63920	0,000000	0,6391	0,00010
66	0,6493	0,6493	0,6493	0,6493	0,6493	0,64930	0,000000	0,6492	0,00010
67	0,6593	0,6593	0,6593	0,6593	0,6593	0,65930	0,000000	0,6592	0,00010
68	0,6692	0,6692	0,6692	0,6692	0,6692	0,66920	0,000000	0,6692	0,00000
							σ_R		σ_E
							0,000014		0,000100

Tabela 5-10: Medidas e estimativas dos experimentos com imagens a 4,7 °com a vertical.

α = 45,9°

T	d_1	d_2	d_3	d_4	d_5	d	s	d_{REF}	e
1	-0,0004	-0,0004	-0,0004	-0,0004	-0,0004	-0,00040	0,00000	0,0000	0,00040
2	0,0092	0,0092	0,0092	0,0092	0,0093	0,00922	0,00004	0,0096	0,00038
3	0,0191	0,0191	0,0191	0,0191	0,0191	0,01910	0,00000	0,0192	0,00010
4	0,0290	0,0290	0,0290	0,0290	0,0290	0,02900	0,00000	0,0292	0,00020
5	0,0390	0,0389	0,0389	0,0389	0,0389	0,03892	0,00004	0,0392	0,00028
6	0,0490	0,0490	0,0490	0,0490	0,0489	0,04898	0,00004	0,0491	0,00012
7	0,0590	0,0591	0,0591	0,0590	0,0591	0,05906	0,00005	0,0591	0,00004
8	0,0690	0,0690	0,0690	0,0690	0,0690	0,06900	0,00000	0,0691	0,00010
9	0,0791	0,0791	0,0791	0,0791	0,0790	0,07908	0,00004	0,0790	0,00008
10	0,0891	0,0891	0,0891	0,0891	0,0890	0,08908	0,00004	0,0890	0,00008
11	0,0991	0,0991	0,0991	0,0991	0,0991	0,09910	0,00000	0,0991	0,00000
12	0,1091	0,1091	0,1091	0,1091	0,1092	0,10912	0,00004	0,1090	0,00012
13	0,1191	0,1191	0,1191	0,1191	0,1191	0,11910	0,00000	0,1190	0,00010
14	0,1292	0,1292	0,1292	0,1291	0,1292	0,12918	0,00004	0,1290	0,00018
15	0,1392	0,1392	0,1392	0,1392	0,1392	0,13920	0,00000	0,1390	0,00020
16	0,1492	0,1492	0,1492	0,1492	0,1492	0,14920	0,00000	0,1490	0,00020
17	0,1592	0,1592	0,1593	0,1593	0,1592	0,15924	0,00005	0,1589	0,00034
18	0,1692	0,1692	0,1692	0,1692	0,1692	0,16920	0,00000	0,1690	0,00020
19	0,1793	0,1793	0,1793	0,1793	0,1793	0,17930	0,00000	0,1790	0,00030
20	0,1893	0,1893	0,1893	0,1892	0,1893	0,18928	0,00004	0,1891	0,00018
21	0,1994	0,1994	0,1994	0,1994	0,1994	0,19940	0,00000	0,1992	0,00020
22	0,2093	0,2093	0,2093	0,2094	0,2093	0,20932	0,00004	0,2092	0,00012
23	0,2193	0,2193	0,2193	0,2193	0,2193	0,21930	0,00000	0,2192	0,00010
24	0,2294	0,2294	0,2294	0,2294	0,2294	0,22940	0,00000	0,2292	0,00020
25	0,2393	0,2392	0,2393	0,2393	0,2393	0,23928	0,00004	0,2392	0,00008
26	0,2494	0,2494	0,2494	0,2494	0,2494	0,24940	0,00000	0,2494	0,00000
27	0,2594	0,2594	0,2594	0,2593	0,2594	0,25938	0,00004	0,2593	0,00008
28	0,2693	0,2693	0,2693	0,2693	0,2693	0,26930	0,00000	0,2693	0,00000
29	0,2794	0,2794	0,2794	0,2794	0,2794	0,27940	0,00000	0,2793	0,00010
30	0,2893	0,2894	0,2893	0,2893	0,2893	0,28932	0,00004	0,2894	0,00008
31	0,2995	0,2994	0,2994	0,2994	0,2995	0,29944	0,00005	0,2995	0,00006
32	0,3094	0,3094	0,3094	0,3093	0,3093	0,30936	0,00005	0,3093	0,00006
33	0,3193	0,3192	0,3192	0,3193	0,3193	0,31926	0,00005	0,3192	0,00006
34	0,3293	0,3293	0,3293	0,3293	0,3293	0,32930	0,00000	0,3293	0,00000
35	0,3393	0,3392	0,3393	0,3392	0,3393	0,33926	0,00005	0,3394	0,00014
36	0,3492	0,3493	0,3493	0,3493	0,3493	0,34928	0,00004	0,3493	0,00002
37	0,3592	0,3592	0,3592	0,3592	0,3592	0,35920	0,00000	0,3592	0,00000
38	0,3691	0,3691	0,3691	0,3691	0,3691	0,36910	0,00000	0,3692	0,00010
39	0,3791	0,3792	0,3792	0,3791	0,3791	0,37914	0,00005	0,3792	0,00006
40	0,3892	0,3891	0,3892	0,3892	0,3891	0,38916	0,00005	0,3892	0,00004
41	0,3993	0,3993	0,3993	0,3993	0,3993	0,39930	0,00000	0,3992	0,00010
42	0,4092	0,4093	0,4092	0,4092	0,4092	0,40922	0,00004	0,4092	0,00002
43	0,4192	0,4192	0,4192	0,4192	0,4192	0,41920	0,00000	0,4191	0,00010
44	0,4292	0,4291	0,4291	0,4292	0,4292	0,42916	0,00005	0,4291	0,00006
45	0,4391	0,4391	0,4391	0,4391	0,4391	0,43910	0,00000	0,4392	0,00010
46	0,4491	0,4491	0,4491	0,4491	0,4491	0,44910	0,00000	0,4492	0,00010
47	0,4591	0,4591	0,4591	0,4591	0,4591	0,45910	0,00000	0,4592	0,00010
48	0,4690	0,4690	0,4691	0,4691	0,4691	0,46906	0,00005	0,4692	0,00014
49	0,4791	0,4791	0,4791	0,4791	0,4791	0,47910	0,00000	0,4792	0,00010
50	0,4890	0,4890	0,4891	0,4890	0,4890	0,48902	0,00004	0,4893	0,00028
51	0,4992	0,4992	0,4992	0,4992	0,4992	0,49920	0,00000	0,4993	0,00010
							σ_R		σ_E
							0,000024		0,000096

Tabela 5-11: Medidas e estimativas dos experimentos com imagens a 45,9 °com a vertical.

				α = 87,2°					
T	d_1	d_2	d_3	d_4	d_5	d	s	d_{REF}	e
1	-0,0004	-0,0004	-0,0004	-0,0004	-0,0004	-0,00040	0,00000	0,0000	0,00040
2	0,0093	0,0092	0,0092	0,0093	0,0093	0,00926	0,00005	0,0096	0,00034
3	0,0191	0,0190	0,0191	0,0190	0,0191	0,01906	0,00005	0,0192	0,00014
4	0,0290	0,0290	0,0290	0,0290	0,0290	0,02900	0,00000	0,0292	0,00020
5	0,0389	0,0389	0,0390	0,0389	0,0389	0,03892	0,00004	0,0392	0,00028
6	0,0490	0,0490	0,0490	0,0490	0,0490	0,04900	0,00000	0,0491	0,00010
7	0,0590	0,0590	0,0591	0,0590	0,0591	0,05904	0,00005	0,0591	0,00006
8	0,0690	0,0690	0,0690	0,0690	0,0690	0,06900	0,00000	0,0691	0,00010
9	0,0791	0,0791	0,0791	0,0791	0,0791	0,07910	0,00000	0,0790	0,00010
10	0,0891	0,0890	0,0890	0,0891	0,0891	0,08906	0,00005	0,0890	0,00006
11	0,0991	0,0991	0,0991	0,0991	0,0991	0,09910	0,00000	0,0991	0,00000
12	0,1091	0,1091	0,1091	0,1091	0,1091	0,10910	0,00000	0,1090	0,00010
13	0,1191	0,1191	0,1191	0,1191	0,1191	0,11910	0,00000	0,1190	0,00010
14	0,1292	0,1292	0,1292	0,1292	0,1292	0,12920	0,00000	0,1290	0,00020
15	0,1392	0,1392	0,1392	0,1392	0,1392	0,13920	0,00000	0,1390	0,00020
16	0,1492	0,1492	0,1492	0,1492	0,1492	0,14920	0,00000	0,1490	0,00020
17	0,1592	0,1592	0,1592	0,1592	0,1592	0,15920	0,00000	0,1589	0,00030
18	0,1692	0,1692	0,1692	0,1692	0,1692	0,16920	0,00000	0,1690	0,00020
19	0,1793	0,1793	0,1793	0,1793	0,1793	0,17930	0,00000	0,1790	0,00030
20	0,1893	0,1893	0,1893	0,1893	0,1893	0,18930	0,00000	0,1891	0,00020
21	0,1994	0,1994	0,1994	0,1994	0,1994	0,19940	0,00000	0,1992	0,00020
22	0,2093	0,2093	0,2093	0,2093	0,2093	0,20930	0,00000	0,2092	0,00010
23	0,2193	0,2193	0,2193	0,2193	0,2193	0,21930	0,00000	0,2192	0,00010
24	0,2294	0,2294	0,2294	0,2294	0,2294	0,22940	0,00000	0,2292	0,00020
25	0,2393	0,2393	0,2393	0,2393	0,2393	0,23930	0,00000	0,2392	0,00010
26	0,2494	0,2494	0,2494	0,2494	0,2494	0,24940	0,00000	0,2494	0,00000
27	0,2594	0,2594	0,2594	0,2593	0,2593	0,25936	0,00005	0,2593	0,00006
28	0,2693	0,2693	0,2693	0,2693	0,2693	0,26930	0,00000	0,2693	0,00000
29	0,2794	0,2794	0,2794	0,2794	0,2794	0,27940	0,00000	0,2793	0,00010
30	0,2893	0,2893	0,2893	0,2893	0,2893	0,28930	0,00000	0,2894	0,00010
31	0,2994	0,2994	0,2994	0,2994	0,2994	0,29940	0,00000	0,2995	0,00010
32	0,3093	0,3093	0,3093	0,3093	0,3093	0,30930	0,00000	0,3093	0,00000
33	0,3193	0,3193	0,3193	0,3193	0,3193	0,31930	0,00000	0,3192	0,00010
34	0,3293	0,3293	0,3293	0,3293	0,3293	0,32930	0,00000	0,3293	0,00000
35	0,3392	0,3392	0,3392	0,3392	0,3392	0,33920	0,00000	0,3394	0,00020
36	0,3493	0,3493	0,3492	0,3492	0,3493	0,34926	0,00005	0,3493	0,00004
37	0,3592	0,3592	0,3592	0,3592	0,3592	0,35920	0,00000	0,3592	0,00000
38	0,3691	0,3691	0,3691	0,3691	0,3691	0,36910	0,00000	0,3692	0,00010
39	0,3791	0,3791	0,3791	0,3791	0,3792	0,37912	0,00004	0,3792	0,00008
40	0,3892	0,3892	0,3892	0,3892	0,3892	0,38920	0,00000	0,3892	0,00000
41	0,3993	0,3993	0,3993	0,3993	0,3993	0,39930	0,00000	0,3992	0,00010
42	0,4092	0,4092	0,4092	0,4092	0,4092	0,40920	0,00000	0,4092	0,00000
43	0,4192	0,4192	0,4192	0,4192	0,4192	0,41920	0,00000	0,4191	0,00010
44	0,4291	0,4292	0,4292	0,4292	0,4292	0,42918	0,00004	0,4291	0,00008
45	0,4391	0,4391	0,4391	0,4391	0,4390	0,43908	0,00004	0,4392	0,00012
46	0,4491	0,4491	0,4491	0,4491	0,4491	0,44910	0,00000	0,4492	0,00010
47	0,4591	0,4591	0,4591	0,4591	0,4591	0,45910	0,00000	0,4592	0,00010
48	0,4691	0,4691	0,4691	0,4691	0,4691	0,46910	0,00000	0,4692	0,00010
49	0,4791	0,4792	0,4792	0,4792	0,4791	0,47916	0,00005	0,4792	0,00004
50	0,4891	0,4891	0,4891	0,4891	0,4891	0,48910	0,00000	0,4893	0,00020
51	0,4991	0,4992	0,4992	0,4992	0,4992	0,49918	0,00004	0,4993	0,00012
							σ_R		σ_E
							0,000012		0,000093

Tabela 5-12: Medidas e estimativas dos experimentos com imagens a 87,2 °com a vertical.

				α = 88,5°					
T	d_1	d_2	d_3	d_4	d_5	d	s	d_{REF}	e
1	-0,0004	-0,0004	-0,0004	-0,0004	-0,0004	-0,00040	0,00000	0,0000	0,00040
2	0,0093	0,0093	0,0093	0,0093	0,0093	0,00930	0,00000	0,0096	0,00030
3	0,0191	0,0191	0,0191	0,0191	0,0191	0,01910	0,00000	0,0192	0,00010
4	0,0290	0,0290	0,0290	0,0290	0,0290	0,02900	0,00000	0,0292	0,00020
5	0,0389	0,0390	0,0390	0,0389	0,0389	0,03894	0,00005	0,0392	0,00026
6	0,0490	0,0490	0,0490	0,0490	0,0489	0,04898	0,00004	0,0491	0,00012
7	0,0590	0,0590	0,0590	0,0591	0,0590	0,05902	0,00004	0,0591	0,00008
8	0,0691	0,0690	0,0690	0,0690	0,0690	0,06902	0,00004	0,0691	0,00008
9	0,0791	0,0791	0,0791	0,0791	0,0791	0,07910	0,00000	0,0790	0,00010
10	0,0890	0,0891	0,0891	0,0891	0,0890	0,08906	0,00005	0,0890	0,00006
11	0,0991	0,0991	0,0991	0,0991	0,0991	0,09910	0,00000	0,0991	0,00000
12	0,1091	0,1091	0,1091	0,1091	0,1091	0,10910	0,00000	0,1090	0,00010
13	0,1191	0,1191	0,1191	0,1191	0,1191	0,11910	0,00000	0,1190	0,00010
14	0,1292	0,1292	0,1292	0,1292	0,1292	0,12920	0,00000	0,1290	0,00020
15	0,1392	0,1391	0,1391	0,1392	0,1392	0,13916	0,00005	0,1390	0,00016
16	0,1492	0,1492	0,1493	0,1492	0,1492	0,14922	0,00004	0,1490	0,00022
17	0,1592	0,1593	0,1592	0,1592	0,1592	0,15922	0,00004	0,1589	0,00032
18	0,1692	0,1692	0,1692	0,1692	0,1692	0,16920	0,00000	0,1690	0,00020
19	0,1793	0,1792	0,1792	0,1793	0,1793	0,17926	0,00005	0,1790	0,00026
20	0,1893	0,1893	0,1893	0,1893	0,1893	0,18930	0,00000	0,1891	0,00020
21	0,1994	0,1994	0,1994	0,1994	0,1994	0,19940	0,00000	0,1992	0,00020
22	0,2093	0,2093	0,2093	0,2094	0,2093	0,20932	0,000042	0,2092	0,00012
23	0,2193	0,2193	0,2193	0,2193	0,2193	0,21930	0,00000	0,2192	0,00010
24	0,2293	0,2294	0,2294	0,2293	0,2294	0,22936	0,00005	0,2292	0,00016
25	0,2393	0,2393	0,2393	0,2393	0,2393	0,23930	0,00000	0,2392	0,00010
26	0,2494	0,2493	0,2493	0,2494	0,2494	0,24936	0,00005	0,2494	0,00004
27	0,2594	0,2593	0,2593	0,2593	0,2594	0,25934	0,00005	0,2593	0,00004
28	0,2693	0,2693	0,2693	0,2693	0,2693	0,26930	0,00000	0,2693	0,00000
29	0,2794	0,2793	0,2793	0,2794	0,2794	0,27936	0,00005	0,2793	0,00006
30	0,2893	0,2893	0,2893	0,2893	0,2893	0,28930	0,00000	0,2894	0,00010
31	0,2994	0,2994	0,2995	0,2994	0,2995	0,29944	0,00005	0,2995	0,00006
32	0,3093	0,3094	0,3093	0,3093	0,3093	0,30932	0,00004	0,3093	0,00002
33	0,3192	0,3193	0,3192	0,3193	0,3192	0,31924	0,00005	0,3192	0,00004
34	0,3293	0,3293	0,3293	0,3293	0,3293	0,32930	0,00000	0,3293	0,00000
35	0,3393	0,3392	0,3392	0,3392	0,3392	0,33922	0,00004	0,3394	0,00018
36	0,3492	0,3493	0,3493	0,3492	0,3492	0,34924	0,00005	0,3493	0,00006
37	0,3592	0,3592	0,3592	0,3592	0,3592	0,35920	0,00000	0,3592	0,00000
38	0,3691	0,3691	0,3691	0,3691	0,3691	0,36910	0,00000	0,3692	0,00010
39	0,3792	0,3791	0,3791	0,3791	0,3792	0,37914	0,00005	0,3792	0,00006
40	0,3891	0,3892	0,3892	0,3891	0,3892	0,38916	0,00005	0,3892	0,00004
41	0,3993	0,3993	0,3993	0,3993	0,3993	0,39930	0,00000	0,3992	0,00010
42	0,4092	0,4092	0,4092	0,4092	0,4092	0,40920	0,00000	0,4092	0,00000
43	0,4192	0,4192	0,4192	0,4192	0,4192	0,41920	0,00000	0,4191	0,00010
44	0,4292	0,4292	0,4292	0,4292	0,4292	0,42920	0,00000	0,4291	0,00010
45	0,4391	0,4391	0,4391	0,4391	0,4391	0,43910	0,00000	0,4392	0,00010
46	0,4491	0,4491	0,4491	0,4491	0,4491	0,44910	0,00000	0,4492	0,00010
47	0,4591	0,4591	0,4591	0,4591	0,4591	0,45910	0,00000	0,4592	0,00010
48	0,4691	0,4691	0,4691	0,4691	0,4691	0,46910	0,00000	0,4692	0,00010
49	0,4791	0,4792	0,4792	0,4791	0,4791	0,47914	0,00005	0,4792	0,00006
50	0,4890	0,4891	0,4891	0,4891	0,4890	0,48906	0,00005	0,4893	0,00024
51	0,4992	0,4991	0,4992	0,4992	0,4992	0,49918	0,00004	0,4993	0,00012
							σ_R		σ_E
							0,000024		0,000088

Tabela 5-13: Medidas e estimativas dos experimentos com imagens a 88,5 °com a vertical.

				α = 89,8°					
T	d_1	d_2	d_3	d_4	d_5	d	s	d_{REF}	e
1	-0,0004	-0,0004	-0,0004	-0,0003	-0,0004	-0,00038	0,00004	0,0000	0,00038
2	0,0093	0,0093	0,0093	0,0092	0,0094	0,00930	0,00007	0,0096	0,00030
3	0,0193	0,0192	0,0190	0,0190	0,0191	0,01912	0,00013	0,0192	0,00008
4	0,0290	0,0290	0,0292	0,0289	0,0292	0,02906	0,00013	0,0292	0,00014
5	0,0388	0,0388	0,0391	0,0391	0,0390	0,03896	0,00015	0,0392	0,00024
6	0,0490	0,0491	0,0489	0,0490	0,0489	0,04898	0,00008	0,0491	0,00012
7	0,0591	0,0592	0,0590	0,0589	0,0592	0,05908	0,00013	0,0591	0,00002
8	0,0690	0,0689	0,0690	0,0690	0,0691	0,06900	0,00007	0,0691	0,00010
9	0,0789	0,0790	0,0791	0,0791	0,0790	0,07902	0,00008	0,0790	0,00002
10	0,0890	0,0892	0,0889	0,0891	0,0890	0,08904	0,00011	0,0890	0,00004
11	0,0991	0,0991	0,0991	0,0991	0,0991	0,09910	0,00000	0,0991	0,00000
12	0,1090	0,1091	0,1092	0,1090	0,1091	0,10908	0,00008	0,1090	0,00008
13	0,1190	0,1190	0,1190	0,1192	0,1190	0,11904	0,00009	0,1190	0,00004
14	0,1293	0,1293	0,1291	0,1292	0,1291	0,12920	0,00010	0,1290	0,00020
15	0,1393	0,1392	0,1393	0,1390	0,1393	0,13922	0,00013	0,1390	0,00022
16	0,1492	0,1492	0,1492	0,1492	0,1492	0,14920	0,00000	0,1490	0,00020
17	0,1592	0,1592	0,1591	0,1594	0,1591	0,15920	0,00012	0,1589	0,00030
18	0,1693	0,1693	0,1692	0,1691	0,1693	0,16924	0,00009	0,1690	0,00024
19	0,1792	0,1792	0,1794	0,1792	0,1793	0,17926	0,00009	0,1790	0,00026
20	0,1892	0,1892	0,1893	0,1894	0,1892	0,18926	0,00009	0,1891	0,00016
21	0,1994	0,1994	0,1995	0,1994	0,1994	0,19942	0,00004	0,1992	0,00022
22	0,2095	0,2094	0,2093	0,2093	0,2094	0,20938	0,00008	0,2092	0,00018
23	0,2193	0,2192	0,2194	0,2193	0,2193	0,21930	0,00007	0,2192	0,00010
24	0,2293	0,2293	0,2294	0,2295	0,2293	0,22936	0,00009	0,2292	0,00016
25	0,2393	0,2393	0,2391	0,2393	0,2392	0,23924	0,00009	0,2392	0,00004
26	0,2494	0,2494	0,2493	0,2494	0,2494	0,24938	0,00004	0,2494	0,00002
27	0,2593	0,2592	0,2595	0,2593	0,2594	0,25934	0,00011	0,2593	0,00004
28	0,2692	0,2692	0,2693	0,2694	0,2691	0,26924	0,00011	0,2693	0,00006
29	0,2795	0,2795	0,2793	0,2793	0,2794	0,27940	0,00010	0,2793	0,00010
30	0,2895	0,2894	0,2894	0,2892	0,2894	0,28938	0,00011	0,2894	0,00002
31	0,2995	0,2995	0,2994	0,2994	0,2994	0,29944	0,00005	0,2995	0,00006
32	0,3092	0,3092	0,3093	0,3094	0,3092	0,30926	0,00009	0,3093	0,00004
33	0,3193	0,3193	0,3191	0,3193	0,3192	0,31924	0,00009	0,3192	0,00004
34	0,3295	0,3294	0,3293	0,3292	0,3294	0,32936	0,00011	0,3293	0,00006
35	0,3392	0,3391	0,3394	0,3392	0,3393	0,33924	0,00011	0,3394	0,00016
36	0,3492	0,3492	0,3492	0,3493	0,3492	0,34922	0,00004	0,3493	0,00008
37	0,3592	0,3593	0,3591	0,3593	0,3591	0,35920	0,00010	0,3592	0,00000
38	0,3692	0,3692	0,3690	0,3691	0,3691	0,36912	0,00008	0,3692	0,00008
39	0,3792	0,3791	0,3792	0,3791	0,3793	0,37918	0,00008	0,3792	0,00002
40	0,3891	0,3890	0,3892	0,3893	0,3891	0,38914	0,00011	0,3892	0,00006
41	0,3993	0,3992	0,3994	0,3993	0,3993	0,39930	0,00007	0,3992	0,00010
42	0,4093	0,4093	0,4092	0,4091	0,4093	0,40924	0,00009	0,4092	0,00004
43	0,4191	0,4191	0,4193	0,4191	0,4193	0,41918	0,00011	0,4191	0,00008
44	0,4291	0,4291	0,4292	0,4292	0,4291	0,42914	0,00005	0,4291	0,00004
45	0,4390	0,4390	0,4390	0,4391	0,4390	0,43902	0,00004	0,4392	0,00018
46	0,4492	0,4491	0,4491	0,4491	0,4491	0,44912	0,00004	0,4492	0,00008
47	0,4591	0,4591	0,4592	0,4590	0,4592	0,45912	0,00008	0,4592	0,00008
48	0,4689	0,4690	0,4692	0,4691	0,4690	0,46904	0,00011	0,4692	0,00016
49	0,4791	0,4791	0,4791	0,4792	0,4791	0,47912	0,00004	0,4792	0,00008
50	0,4892	0,4892	0,4890	0,4890	0,4892	0,48912	0,00011	0,4893	0,00018
51	0,4991	0,4993	0,4991	0,4992	0,4992	0,49918	0,00008	0,4993	0,00012
							σ_R		σ_E
							0,000086		0,000088

Tabela 5-14: Medidas e estimativas dos experimentos com imagens a 89,8 °com a vertical.

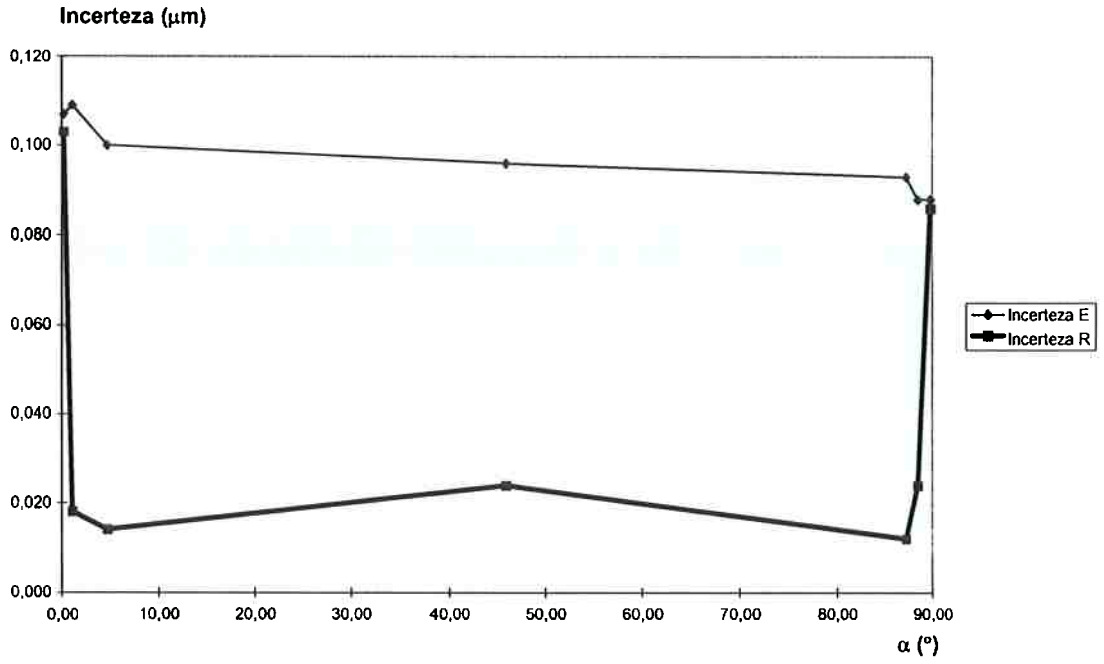

Figura 5.6: Exatidão e repetibilidade do IMVC versus posição angular α do objeto no plano-imagem.

É imediato observar no gráfico acima que: 1º) A exatidão das medidas aumenta gradualmente quando as retas da imagem evoluem da posição vertical à horizontal, ocorrendo um pequeno decréscimo próximo às condições de alinhamento exato com aquelas direções; 2º) Nestas particulares configurações, o processo de medição apresenta o pior desempenho relativamente à repetibilidade, porém é suficiente um pequeno desalinhamento das retas para que haja uma abrupta diminuição na incerteza das medidas; 3º) Próximo à orientação de 45°, a repetibilidade sofre um novo decréscimo.

Esses resultados são conseqüência da interação de dois fenômenos que ocorrem no processo de aquisição da imagem: filtragem opto-eletrônica da imagem focalizada ideal e quantização espacial do sinal digitalizado.

No primeiro caso, a aplicação de filtro *anti-aliasing* às sucessivas linhas do sinal de vídeo torna a *PSF* do sistema levemente assimétrica, apresentando relação $f_H / f_V < 1$ entre as freqüências de corte ao longo da horizontal e vertical, respectivamente. Dessa

forma, gradientes de nível de cinza na direção vertical se aproximam mais do modelo *degrau* do que ao longo da horizontal, o que explica as medidas mais exatas obtidas para imagens cujas retas são aproximadamente horizontais. Não é demais observar ainda que essa assimetria direcional já poderia ter sido identificada através da simples comparação das imagens da figura 5.4, onde se nota que os traços verticais apresentam-se nitidamente esmaecidos quando comparados com os horizontais.

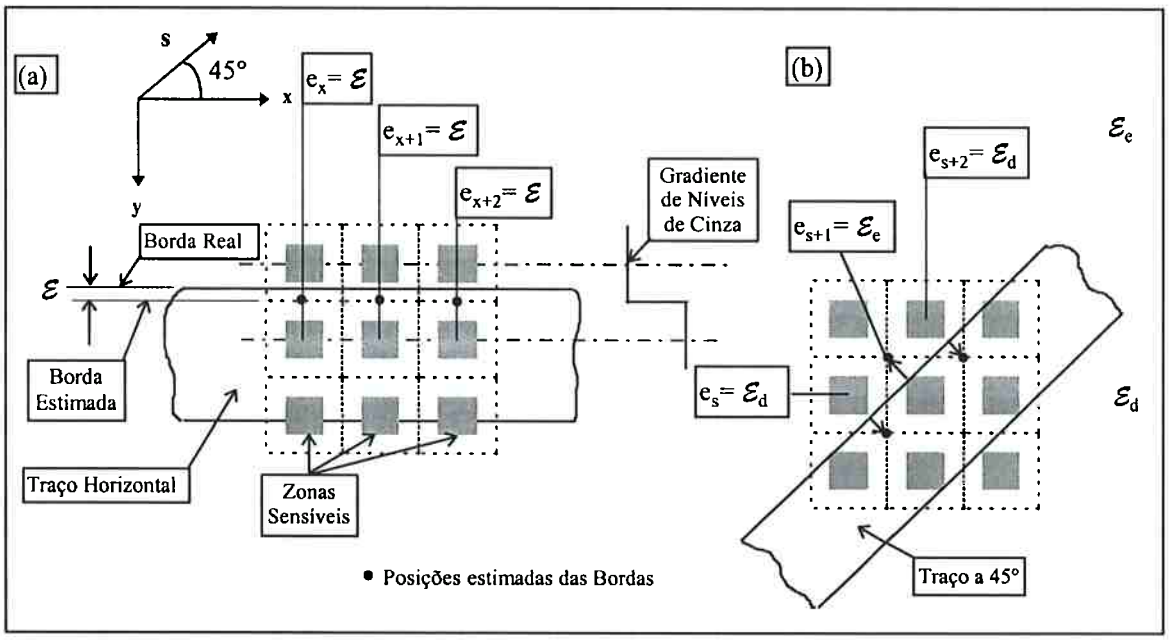

Figura 5.7: Efeito da quantização espacial da imagem sobre a incerteza das medidas: (a) Traço alinhado com a horizontal; (b) Traço alinhado com a diagonal.

A perda de repetibilidade ao longo dos ângulos de 0°, 90° e 45° é causada pelo abrupto aumento da correlação entre os erros das medidas dos sensores do CCD quando da ocorrência de alinhamento entre os traços do objeto e aquelas particulares direções. Conforme mostrado na figura 5.7, o erro de localização de um pixel P se reproduz ao longo de toda a seqüência de pixels p_i alinhados com P, viesando significativamente as medidas. A 0 e 90°, pelo fato de o erro ter magnitude e sinal sempre iguais e replicar-se

pixel a pixel, esse viés é sensivelmente maior que ao longo da diagonal, onde um padrão seqüencial de erros de sinais opostos $\{\varepsilon_e, \varepsilon_d\}$ se alterna a cada intervalo de $\sqrt{2}$ pixels.

Os resultados e considerações anteriores permitem estabelecer que, para que se obtenha a melhor relação de compromisso entre exatidão e repetibilidade, os objetos a serem medidos devem estar orientados no plano-imagem segundo uma particular direção; no caso da microescala, seus traços devem estar inclinados de ± 2 a 3° com a horizontal, e para a microtextura de peneiras, contendo pares ortogonais de traços, estes devem estar inclinados de ângulos em torno de ± 45°.

5.5 Medição do Desempenho dos Instrumentos Metrológicos para Microescalas

Os experimentos de ajuste de parâmetros e avaliação dos IMVC's dedicados à calibração de microescalas foram baseados em amostras de 5 imagens de microescalas, capturadas nas condições expostas no ítem 5.3, com as seguintes configurações: 1º) traços a 89,8° com a vertical (figura 5.5.g) para o IMVC-E_1; 2º) traços a 88,5° (figura 5.5.f) para a série IMVC-E_2 a IMVC-E_5.

Dos referidos experimentos, são extraídas as seguintes informações:

a. posições d_i dos traços, seus valores médios $\overline{d}_i$, desvios-padrão s_i e desvios e_i em relação às posições d_{REF} de referência;
b. média σ_R de s_i e média quadrática σ_E de e_i;
c. incerteza dos valores médios $\overline{d}_i$ para um nível de significância de 95%, ou seja:

$$\delta d_i = t_{4;0,05} \cdot \frac{s_i}{\sqrt{5}}, \tag{5-6}$$

onde $t_{4;0,05} = 2{,}776$ é o valor correspondente a 95% da área da distribuição t de Student para amostras com 4 graus de liberdade.

5.4.1 Medição do Desempenho do IMVC-E_1

Como esse instrumento recupera apenas uma informação global da microescala — a freqüência f_0 de repetição dos traços (em *nº de pixels*) — sua aplicação a ensaios de calibração tem carácter eminentemente qualitativo. Obtido f_0, pode-se atribuir um índice de qualidade Q_T a uma nova microescala M_T e freqüência espacial f_0^T utilizando o padrão de qualidade Q_{REF} de uma microescala-referência M_{REF}[3] e freqüência f_0^{REF}, ou seja:

$$\begin{aligned} Q_T = Q_{REF} &\Leftrightarrow f_0^T \in \left\{ f_0^{REF} - \delta f, f_0^{REF} + \delta f \right\} \\ Q_T < Q_{REF} &\Leftrightarrow f_0^T \notin \left\{ f_0^{REF} - \delta f, f_0^{REF} + \delta f \right\} \end{aligned} \qquad (5\text{-}7)$$

onde o máximo padrão de qualidade possível Q_{REF} corresponde à microescala ideal, com traços equiespaçados de 0,01mm[4] e o valor δf é a incerteza da estimativa de f_0^{REF}.

Dado que o IMVC-E_1 não possui parâmetros a serem previamente ajustados, o seu desempenho foi medido diretamente a partir de experimentos realizados com o grupo de 5 imagens da série 89,8°, os resultados sendo reportados na tabela 5-15.

Imagem	f_0
1	9,485
2	9,442
3	9,480
4	9,419
5	9,496
f_0 médio	9,4644
σ_R	0,0325
δf	0,0404

(nº de pixels)

Tabela 5-15: Repetibilidade do IMVC-E_1.

Notando-se que, para a constante de amplificação geométrica adotada, 1 *pixel* corresponde a, aproximadamente 1 μm, os resultados obtidos revelam que o IMVC-E_1

[3] O assim chamado "golden template", para se manter fiel à nomenclatura de inspeção visual automática.
[4] No ensaio realizado, o valor estimado de f_0 corresponde a uma microescala com traços equiespaçados de 0.009986 mm.

possui elevada repetibilidade, fornecendo valores de f_0 dentro de um intervalo da ordem de ±0,00004 μm !

Em um ensaio típico utilizando esse instrumento é suficiente que as espessuras das pastilhas onde são impressas as microescalas teste M^T e referência M^{REF} estejam dentro de uma tolerância de 0,06mm [5] para que o IMVC-E_1, após determinar o valor médio da freqüência espacial f_0^T, atribua a M^T um índice de qualidade $Q^T = Q^{REF}$ caso $9{,}424 \leq f_0^T \leq 9{,}505$, ou $Q^T < Q^{REF}$ caso f_0^T esteja fora deste intervalo.

5.4.2 Medição do Desempenho do IMVC-E_2

Esse instrumento, baseado na aplicação direta da transformada de Hough à imagem em tons de cinza da microescala, apresenta desempenho metrológico bastante variável em função dos valores de seus parâmetros, que determinam, simultaneamente: forma e quantização do espaço de Hough, largura de seu espectro de freqüências, exatidão do método de localização de picos; e, em casos extremos, podem induzir detecção de falsos picos e exclusão de picos verdadeiros (figura 5.8).

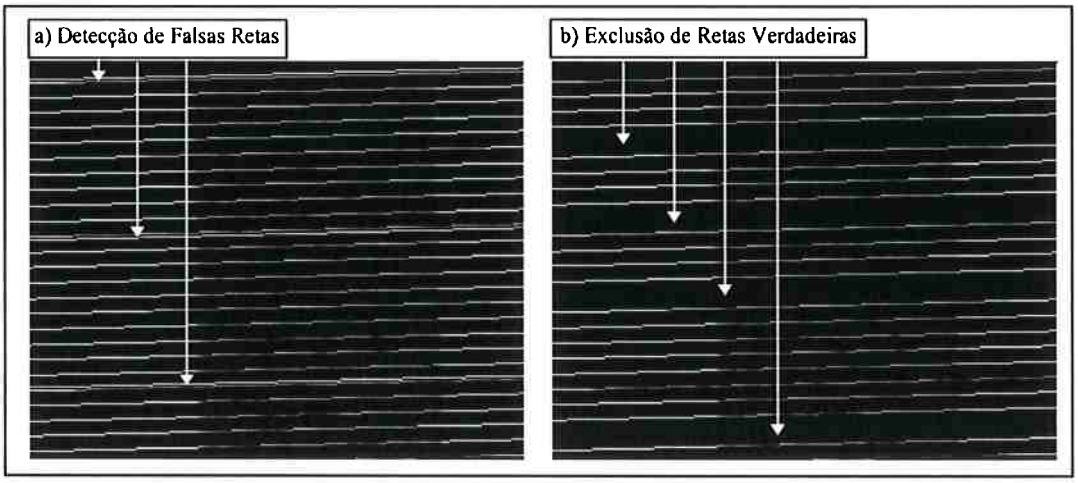

Figura 5.8: Falhas graves do IMVC-E_2*: (a) Detecção de falsas retas; (b) Perda de retas verdadeiras.*

[5] Se esse não fôr o caso, pode-se compensar a diferença através de blocos-padrão.

Para um operador metrologista, a atribuição de valores a esses parâmetros seria uma tarefa bastante difícil, o que torna necessário os experimentos para identificação, *a priori*, de condições operacionais adequadas para o instrumento.

Considerando-se que a orientação ideal dos traços da imagem, conforme exposto no ítem 5.3, é da ordem de 2 a 3 graus com a direção horizontal, estabeleceu-se, para os parâmetros θ_{min}, θ_{max}, os valores 86° e 90°, respectivamente. Para o limiar L_h, adotou-se o valor 0,4, pouco inferior ao quociente ($L_{min}/L_{max} \cong 0{,}5$) entre os comprimentos extremos dos traços. Para o incremento angular $d\theta$, foram testados 4 valores: 1,00°, 0,50°, 0,25° e 0,125°. L_{min}, apesar de valer cerca de 150 pixels, foi testado com os valores 100, 150 e 200, e a meia-espessura E, com os valores 1 e 2. Finalmente, para a largura s_ρ da zona de espalhamento dos picos da transformada foram arbitrados 3 valores: 4, 6 e 8.

$d\theta$			1.000°	0.500°	0.250°	0.125°
Sρ	Lmin	E	σ_E	σ_E	σ_E	σ_E
4	100	1	0,252	0,132	0,138	0,183
4	100	2	0,252	0,132	0,138	0,183
4	150	1	0,173	0,191	0,130	0,153
4	150	2	0,173	0,191	0,130	0,138
4	200	1	0,318	0,201	0,119	0,138
4	200	2	0,318	0,153	0,119	0,138
6	100	1	0,139	0,139	0,138	0,183
6	100	2	0,159	0,132	0,138	0,183
6	150	1	0,138	0,096	0,143	0,153
6	150	2	0,158	0,129	0,130	0,153
6	200	1	0,149	0,129	0,139	0,138
6	200	2	0,179	0,159	0,119	0,138
8	100	1	0,153	0,148	0,145	0,183
8	100	2	0,139	0,140	0,138	0,183
8	150	1	0,226	0,172	0,137	0,184
8	150	2	0,198	0,143	0,128	0,153
8	200	1	0,162	0,154	0,150	0,174
8	200	2	0,229	0,139	0,140	0,142

Tabela 5-16: Desempenho do IMVC-E_2 *em função de* ($d\theta$, L_{min}, E, s_ρ).

Utilizando-se uma única imagem da microescala da série 88,5°, realizaram-se 72 experimentos de avaliação do desempenho do IMVC-E_2, combinando-se os 4 parâmetros arbitrados acima. Como antes, foram determinadas as distâncias d_i, os desvios e_i e a média quadrática σ_E desses desvios. Na tabela 5-16, apresenta-se a súmula das medidas

experimentais, através do registro da estimativa σ_E da exatidão do IMVC-E_2 nos pontos $(d\theta, L_{min}, E, s_\rho)$.

Como bem o demonstra a análise de variança de σ_E em função de cada um dos parâmetros, $d\theta$ e s_ρ são os que mais influem na exatidão das medidas, já que, a nível de 95% de significância, cada qual, isoladamente, modifica o valor de σ_E (tabelas 5-17 e 5-18), o mesmo não ocorrendo com L_{min} e E, cuja influência só é observada fixados os demais parâmetros (tabelas 5-19 e 5-20).

ANOVA: fator único (dθ)

Hipótese H0: $\sigma_E^{1.000} = \sigma_E^{0.500} = \sigma_E^{0.250} = \sigma_E^{0.125}$

Grupo	*Contagem*	*Soma*	*Média*	*Variância*		
1.000	18	3,51426302	0,19523683	0,00335858		
0.500	18	2,67915540	0,14884197	0,00069053		
0.250	18	2,41958007	0,13442112	0,00007869		
0.125	18	2,90057950	0,16114331	0,00041245		
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Entre grupos	0,03639044	3	0,01213015	10,6867707	7,6702E-06	2,73949752
Dentro dos grupos	0,0771842	68	0,00113506			
Total	0,11357464	71				

Conclusão: $F > F_{crítico} \Rightarrow \sigma_E^{1.000} \neq \sigma_E^{0.500} \neq \sigma_E^{0.250} \neq \sigma_E^{0.125}$

Tabela 5-17: Influência isolada de $d\theta$ sobre σ_E.

ANOVA: fator único (sρ)

Hipótese H0: $\sigma_E^4 = \sigma_E^6 = \sigma_E^8$

Grupo	*Contagem*	*Soma*	*Média*	*Variância*		
4	24	4,19296353	0,17470681	0,00329249		
6	24	3,46044151	0,14418506	0,00039783		
8	24	3,86017296	0,16084054	0,00076031		
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Entre grupos	0,01121005	2	0,00560502	3,77812844	0,02773063	3,12964232
Dentro dos grupos	0,10236459	69	0,00148354			
Total	0,11357464	71				

Conclusão: $F > F_{crítico} \Rightarrow \sigma_E^4 \neq \sigma_E^6 \neq \sigma_E^8$

Tabela 5-18: Influência isolada de s_ρ sobre σ_E.

ANOVA: fator único (L_{min})

Hipótese H0: $\sigma_E^{100} = \sigma_E^{150} = \sigma_E^{200}$

Grupo	*Contagem*	*Soma*	*Média*	*Variância*		
100	24	3,84991031	0,16041293	0,00117686		
150	24	3,72083645	0,15503485	0,00083357		
200	24	3,94283124	0,16428464	0,00288256		
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Entre grupos	0,00103578	2	0,00051789	0,31752903	0,72900353	3,12964232
Dentro dos grupos	0,11253886	69	0,001631			
Total	0,11357464	71				

Conclusão: $F<F_{crítico} \Rightarrow \sigma_E^{100} = \sigma_E^{150} = \sigma_E^{200}$

Tabela 5-19: Ausência de influência isolada de L_{min} sobre σ_E.

ANOVA: fator único (E)

Hipótese H0: $\sigma_E^{1} = \sigma_E^{2}$

Grupo	*Contagem*	*Soma*	*Média*	*Variância*		
1	36	5,79994609	0,16110961	0,00162442		
2	36	5,71363191	0,15871200	0,00161761		
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Entre grupos	0,00010347	1	0,00010347	0,06383286	0,80127785	3,97778877
Dentro dos grupos	0,11347117	70	0,00162102			
Total	0,11357464	71				

Conclusão: $F<F_{crítico} \Rightarrow \sigma_E^{1} = \sigma_E^{2}$

Tabela 5-20: Ausência de influência isolada de E sobre σ_E.

Analisando-se a tabela 5-18, adota-se s_ρ=6, por ser esta a condição que minimiza a esperança de σ_E, independentemente dos demais fatores. Em seguida, adota-se $d\theta$ = 0,5°, pois esse incremento angular minimiza a esperança da distribuição $\sigma_E(d\theta | s_\rho=6)$ (ver tabela 5-21). As demais escolhas — L_{min}= 150 e E = 1 — são quase mandatórias, uma vez que valores alternativos acarretam sensível aumento de σ_E.

dθ		1.000	0.500	0.250	0.125
Lmin	E	σ_E	σ_E	σ_E	σ_E
100	1	0,139	0,139	0,138	0,183
100	2	0,159	0,132	0,138	0,183
150	1	0,138	0,096	0,143	0,153
150	2	0,158	0,129	0,130	0,153
200	1	0,149	0,129	0,139	0,138
200	2	0,179	0,159	0,119	0,138
$E\{\sigma_E\}$		0,1537	0,1305	0,1345	0,1580

Tabela 5-21: Valores médios para a distribuição $\sigma_E(d\theta / s_\rho=6)$.

Na tabela 5-22, mostra-se ainda que pequenas variações segundo $d\theta$ e s_ρ , em torno do ponto ($d\theta$=0,50°,s_ρ=6,L_{min}=150,E=1), não diminuem o valor de σ_E, de modo que esse ponto será considerado como a condição própria de operação do IMVC-E_2.

		sρ=5			sρ=6			sρ=7	
T\dθ	0.75	0.50	0.375	0.75	0.50	0.375	0.75	0.50	0.375
1	0,0007	0,0006	0,0005	0,0003	0,0000	0,0003	0,0004	0,0001	0,0005
2	0,0007	0,0007	0,0003	0,0002	0,0003	0,0004	0,0003	0,0000	0,0004
3	0,0002	0,0000	0,0001	0,0001	0,0002	0,0000	0,0002	0,0004	0,0001
4	0,0001	0,0003	0,0002	0,0004	0,0004	0,0005	0,0000	0,0001	0,0002
5	0,0004	0,0001	0,0003	0,0000	0,0004	0,0004	0,0003	0,0003	0,0005
6	0,0005	0,0002	0,0003	0,0002	0,0003	0,0002	0,0005	0,0004	0,0002
7	0,0000	0,0002	0,0003	0,0004	0,0002	0,0002	0,0002	0,0001	0,0000
8	0,0002	0,0001	0,0001	0,0001	0,0002	0,0001	0,0000	0,0001	0,0003
9	0,0004	0,0004	0,0001	0,0002	0,0001	0,0001	0,0002	0,0000	0,0001
10	0,0002	0,0000	0,0000	0,0002	0,0000	0,0001	0,0001	0,0002	0,0002
11	0,0001	0,0003	0,0002	0,0002	0,0001	0,0001	0,0003	0,0000	0,0001
12	0,0004	0,0001	0,0002	0,0003	0,0000	0,0003	0,0001	0,0004	0,0003
13	0,0002	0,0002	0,0004	0,0000	0,0001	0,0004	0,0003	0,0001	0,0004
14	0,0000	0,0002	0,0003	0,0004	0,0001	0,0005	0,0001	0,0001	0,0001
15	0,0005	0,0006	0,0002	0,0001	0,0001	0,0000	0,0003	0,0002	0,0003
16	0,0002	0,0002	0,0001	0,0005	0,0002	0,0000	0,0000	0,0000	0,0005
17	0,0001	0,0000	0,0001	0,0003	0,0003	0,0002	0,0004	0,0004	0,0003
18	0,0005	0,0003	0,0005	0,0000	0,0002	0,0002	0,0000	0,0000	0,0004
19	0,0003	0,0000	0,0004	0,0003	0,0003	0,0003	0,0003	0,0003	0,0001
20	0,0001	0,0003	0,0002	0,0000	0,0002	0,0002	0,0004	0,0000	0,0002
21	0,0003	0,0001	0,0001	0,0003	0,0002	0,0002	0,0001	0,0002	0,0003
22	0,0001	0,0002	0,0001	0,0000	0,0002	0,0003	0,0003	0,0005	0,0000
23	0,0002	0,0001	0,0002	0,0004	0,0002	0,0002	0,0000	0,0002	0,0003
24	0,0004	0,0003	0,0004	0,0001	0,0003	0,0004	0,0003	0,0000	0,0004
25	0,0001	0,0001	0,0002	0,0002	0,0003	0,0001	0,0000	0,0002	0,0001
26	0,0003	0,0001	0,0000	0,0000	0,0001	0,0002	0,0000	0,0001	0,0001
27	0,0003	0,0001	0,0001	0,0001	0,0003	0,0000	0,0004	0,0003	0,0001
28	0,0000	0,0002	0,0002	0,0002	0,0003	0,0000	0,0001	0,0000	0,0001
29	0,0002	0,0001	0,0004	0,0000	0,0002	0,0001	0,0002	0,0002	0,0003
30	0,0002	0,0002	0,0002	0,0002	0,0002	0,0001	0,0001	0,0000	0,0001
31	0,0001	0,0002	0,0000	0,0001	0,0002	0,0000	0,0001	0,0002	0,0000
32	0,0002	0,0003	0,0001	0,0002	0,0002	0,0003	0,0000	0,0001	0,0001
33	0,0004	0,0001	0,0000	0,0002	0,0000	0,0001	0,0002	0,0000	0,0002
34	0,0000	0,0003	0,0001	0,0001	0,0001	0,0001	0,0000	0,0002	0,0002
35	0,0003	0,0001	0,0004	0,0001	0,0002	0,0001	0,0001	0,0001	0,0001
36	0,0002	0,0003	0,0004	0,0000	0,0001	0,0000	0,0001	0,0002	0,0002
37	0,0001	0,0002	0,0002	0,0002	0,0001	0,0002	0,0003	0,0004	0,0000
38	0,0001	0,0002	0,0001	0,0001	0,0001	0,0002	0,0000	0,0001	0,0003
39	0,0004	0,0002	0,0000	0,0001	0,0001	0,0002	0,0002	0,0002	0,0001
40	0,0001	0,0001	0,0001	0,0002	0,0001	0,0002	0,0001	0,0000	0,0001
41	0,0000	0,0002	0,0001	0,0000	0,0002	0,0001	0,0002	0,0002	0,0003
42	0,0004	0,0000	0,0003	0,0003	0,0002	0,0000	0,0001	0,0000	0,0000
43	0,0002	0,0004	0,0003	0,0002	0,0003	0,0001	0,0003	0,0004	0,0001
44	0,0000	0,0001	0,0002	0,0001	0,0002	0,0002	0,0000	0,0002	0,0001
45	0,0004	0,0004	0,0000	0,0004	0,0003	0,0001	0,0004	0,0002	0,0001
46	0,0001	0,0000	0,0001	0,0001	0,0002	0,0003	0,0002	0,0004	0,0002
47	0,0001	0,0003	0,0002	0,0003	0,0002	0,0003	0,0001	0,0001	0,0000
48	0,0004	0,0000	0,0003	0,0000	0,0002	0,0002	0,0002	0,0001	0,0002
49	0,0001	0,0002	0,0004	0,0002	0,0001	0,0001	0,0000	0,0001	0,0000
50	0,0002	0,0000	0,0000	0,0000	0,0002	0,0002	0,0003	0,0004	0,0004
51	0,0003	0,0003	0,0000	0,0002	0,0002	0,0001	0,0001	0,0001	0,0002
	σ_E	σ_E	σ_E	σ_E	σ_E	σ_E	σ_E	σ_E	σ_E
	0,000173	0,000157	0,000141	0,000133	**0,000096**	0,000131	0,000140	0,000142	0,000142

Tabela 5-22: Refinamento das medidas de σ_E em torno do ponto (0,50; 6; 150;1).

A tabela 5-23, finalmente, apresenta as medidas, incertezas e estimativas de exatidão e repetibilidade desse instrumento, obtidas com as 5 imagens da série 88,5°. Conforme se pode observar nessa tabela, a incerteza média do IMVC-E_2 (0,00019mm) é praticamente igual à do ensaio convencional (±0,0002mm).

T	d_1	d_2	d_3	d_4	d_5	d	s	δ	d_{REF}	e
01	0,0000	-0,0005	-0,0002	-0,0001	-0,0002	-0,0002	0,00019	0,00023	0,0000	0,0002
02	0,0093	0,0095	0,0091	0,0092	0,0092	0,0093	0,00015	0,00019	0,0096	0,0003
03	0,0194	0,0188	0,0191	0,0192	0,0192	0,0191	0,00022	0,00027	0,0192	0,0001
04	0,0288	0,0289	0,0292	0,0293	0,0292	0,0291	0,00022	0,00027	0,0292	0,0001
05	0,0388	0,0389	0,0392	0,0387	0,0392	0,0390	0,00023	0,00029	0,0392	0,0002
06	0,0488	0,0489	0,0492	0,0487	0,0486	0,0488	0,00023	0,00029	0,0491	0,0003
07	0,0589	0,0590	0,0593	0,0593	0,0587	0,0590	0,00026	0,00032	0,0591	0,0001
08	0,0689	0,0690	0,0687	0,0694	0,0693	0,0691	0,00029	0,00036	0,0691	0,0000
09	0,0789	0,0790	0,0793	0,0788	0,0794	0,0791	0,00026	0,00032	0,0790	0,0001
10	0,0890	0,0891	0,0893	0,0888	0,0888	0,0890	0,00021	0,00026	0,0890	0,0000
11	0,0990	0,0991	0,0988	0,0995	0,0994	0,0992	0,00029	0,00036	0,0991	0,0001
12	0,1090	0,1091	0,1088	0,1089	0,1095	0,1091	0,00027	0,00034	0,1090	0,0001
13	0,1191	0,1191	0,1194	0,1189	0,1189	0,1191	0,00020	0,00025	0,1190	0,0001
14	0,1291	0,1292	0,1289	0,1296	0,1289	0,1291	0,00029	0,00036	0,1290	0,0001
15	0,1391	0,1392	0,1389	0,1390	0,1396	0,1392	0,00027	0,00034	0,1390	0,0002
16	0,1492	0,1493	0,1495	0,1490	0,1490	0,1492	0,00021	0,00026	0,1490	0,0002
17	0,1592	0,1593	0,1590	0,1591	0,1590	0,1591	0,00013	0,00016	0,1589	0,0002
18	0,1692	0,1693	0,1690	0,1691	0,1691	0,1691	0,00011	0,00014	0,1690	0,0001
19	0,1793	0,1794	0,1791	0,1791	0,1791	0,1792	0,00014	0,00018	0,1790	0,0002
20	0,1893	0,1894	0,1891	0,1892	0,1891	0,1892	0,00013	0,00016	0,1891	0,0001
21	0,1994	0,1994	0,1992	0,1992	0,1992	0,1993	0,00011	0,00014	0,1992	0,0001
22	0,2094	0,2095	0,2092	0,2092	0,2092	0,2093	0,00014	0,00018	0,2092	0,0001
23	0,2194	0,2195	0,2192	0,2193	0,2192	0,2193	0,00013	0,00016	0,2192	0,0001
24	0,2295	0,2295	0,2292	0,2293	0,2293	0,2294	0,00013	0,00017	0,2292	0,0002
25	0,2395	0,2395	0,2393	0,2393	0,2393	0,2394	0,00011	0,00014	0,2392	0,0002
26	0,2495	0,2496	0,2493	0,2494	0,2493	0,2494	0,00013	0,00016	0,2494	0,0000
27	0,2596	0,2591	0,2593	0,2594	0,2594	0,2594	0,00018	0,00023	0,2593	0,0001
28	0,2690	0,2691	0,2693	0,2694	0,2694	0,2692	0,00018	0,00023	0,2693	0,0001
29	0,2791	0,2791	0,2794	0,2794	0,2794	0,2793	0,00016	0,00020	0,2793	0,0000
30	0,2896	0,2891	0,2894	0,2894	0,2894	0,2894	0,00018	0,00022	0,2894	0,0000
31	0,2997	0,2992	0,2995	0,2995	0,2995	0,2995	0,00018	0,00022	0,2995	0,0000
32	0,3091	0,3092	0,3095	0,3095	0,3095	0,3094	0,00019	0,00024	0,3093	0,0001
33	0,3192	0,3192	0,3195	0,3195	0,3190	0,3193	0,00022	0,00027	0,3192	0,0001
34	0,3292	0,3292	0,3295	0,3295	0,3290	0,3293	0,00022	0,00027	0,3293	0,0000
35	0,3392	0,3393	0,3390	0,3390	0,3396	0,3392	0,00025	0,00031	0,3394	0,0002
36	0,3492	0,3493	0,3495	0,3490	0,3490	0,3492	0,00021	0,00026	0,3493	0,0001
37	0,3593	0,3593	0,3590	0,3591	0,3591	0,3592	0,00013	0,00017	0,3592	0,0000
38	0,3693	0,3693	0,3690	0,3691	0,3691	0,3692	0,00013	0,00017	0,3692	0,0000
39	0,3793	0,3788	0,3791	0,3791	0,3791	0,3791	0,00018	0,00022	0,3792	0,0001
40	0,3893	0,3894	0,3891	0,3891	0,3891	0,3892	0,00014	0,00018	0,3892	0,0000
41	0,3994	0,3994	0,3992	0,3992	0,3992	0,3993	0,00011	0,00014	0,3992	0,0001
42	0,4094	0,4095	0,4092	0,4092	0,4092	0,4093	0,00014	0,00018	0,4092	0,0001
43	0,4194	0,4189	0,4192	0,4192	0,4192	0,4192	0,00018	0,00022	0,4191	0,0001
44	0,4289	0,4295	0,4292	0,4292	0,4293	0,4292	0,00022	0,00027	0,4291	0,0001
45	0,4389	0,4389	0,4392	0,4392	0,4393	0,4391	0,00019	0,00023	0,4392	0,0001
46	0,4490	0,4490	0,4493	0,4492	0,4487	0,4490	0,00023	0,00029	0,4492	0,0002
47	0,4590	0,4590	0,4587	0,4593	0,4593	0,4591	0,00025	0,00031	0,4592	0,0001
48	0,4690	0,4690	0,4693	0,4688	0,4693	0,4691	0,00022	0,00027	0,4692	0,0001
49	0,4791	0,4791	0,4794	0,4794	0,4794	0,4793	0,00016	0,00020	0,4792	0,0001
50	0,4891	0,4891	0,4894	0,4888	0,4888	0,4890	0,00025	0,00031	0,4893	0,0003
51	0,4991	0,4991	0,4994	0,4994	0,4994	0,4993	0,00016	0,00020	0,4993	0,0000
							σ_R			σ_E
							0,000191			0,000075

Tabela 5-23: Repetibilidade e exatidão do IMVC-E_2.

5.5.3 Medição do Desempenho do IMVC-E_3

Fixando-se uma condição inicial arbitrária para as medições em escala nominal, quais sejam — *limiarização=automática*, *esqueletonização=ausente* — investigou-se o efeito dos parâmetros das outras duas escalas sobre o resultado das medidas, utilizando-se uma das imagens da série 88,5°.

Analisando-se, inicialmente, o parâmetro *Sigma Unitário*, verificou-se que a alteração [6] no valor do fator de conversão de *nº de pixels* para *mm*, quando se substituem as variâncias σ_i de cada reta ajustada $\mathcal{R}_i$ [7] por um único valor σ=1, é de ordem superior, não causando quaisquer modificações nas medidas d_i.

Com relação ao papel das ferramentas de medição em escala ordinal e racional sobre a exatidão das medidas, a tabela 5-24 indica que a utilização do algoritmo de rotulação tem um impacto desfavorável sobre os resultados. Isso se explica pelo fato de, nesse caso, se ajustarem, a cada conjunto de pixels $\mathcal{T}_i$, uma reta $\mathcal{R}_i$ orientada segundo θ_i, enquanto que, usando a transformada de Hough, identifica-se um único ângulo θ_{feixe} correspondente ao feixe de n retas paralelas, ajustando-se em seguida o conjunto das n retas, conforme descrito no capítulo 3.

Por outro lado, na mesma tabela, o acréscimo em exatidão observado quando se refinam as posições das retas após aplicação da transformada de Hough, deve-se ao fato de que o algoritmo de ajuste exclui os pixels mais distantes dos eixos mediais dos traços, já que considera como pertencentes à reta $\mathcal{R}_i$ apenas os pixels distando 1,5 pixels ou menos da sua posição estimada inicial d_i^0.

---

[6] Da ordem de 0,00002 μm/pixel.

[7] Segundo CAGLIOTI (1993), a incerteza σ_i de retas que se assemelham a segmentos de retas digitais estaria mais próxima de 1 do que do valor estimado pelo método dos mínimos quadrados.

Ord Rac	Rotulação MMQ	Tr. Hough Tr. Hough	Tr. Hough MMQ		$e=\|d-d_{REF}\|$		
T	d_A	d_B	d_C	d_{REF}	e_A	e_B	e_C
01	-0,0005	-0,0006	-0,0005	0,0000	0,0005	0,0006	0,0005
02	0,0095	0,0094	0,0093	0,0096	0,0001	0,0002	0,0003
03	0,0186	0,0193	0,0191	0,0192	0,0006	0,0001	0,0001
04	0,0294	0,0288	0,0290	0,0292	0,0002	0,0004	0,0002
05	0,0384	0,0389	0,0389	0,0392	0,0008	0,0003	0,0003
06	0,0487	0,0489	0,0489	0,0491	0,0004	0,0002	0,0002
07	0,0592	0,0589	0,0590	0,0591	0,0001	0,0002	0,0001
08	0,0688	0,0689	0,0690	0,0691	0,0003	0,0002	0,0001
09	0,0797	0,0790	0,0790	0,0790	0,0007	0,0000	0,0000
10	0,0888	0,0890	0,0891	0,0890	0,0002	0,0000	0,0001
11	0,0990	0,0991	0,0991	0,0991	0,0001	0,0000	0,0000
12	0,1099	0,1091	0,1091	0,1090	0,0009	0,0001	0,0001
13	0,1186	0,1191	0,1192	0,1190	0,0004	0,0001	0,0002
14	0,1289	0,1292	0,1293	0,1290	0,0001	0,0002	0,0003
15	0,1394	0,1392	0,1392	0,1390	0,0004	0,0002	0,0002
16	0,1494	0,1492	0,1492	0,1490	0,0004	0,0002	0,0002
17	0,1598	0,1592	0,1592	0,1589	0,0009	0,0003	0,0003
18	0,1689	0,1693	0,1692	0,1690	0,0001	0,0003	0,0002
19	0,1791	0,1793	0,1793	0,1790	0,0001	0,0003	0,0003
20	0,1902	0,1893	0,1893	0,1891	0,0011	0,0002	0,0002
21	0,1993	0,1994	0,1994	0,1992	0,0001	0,0002	0,0002
22	0,2090	0,2094	0,2093	0,2092	0,0002	0,0002	0,0001
23	0,2201	0,2194	0,2193	0,2192	0,0009	0,0002	0,0001
24	0,2290	0,2295	0,2294	0,2292	0,0002	0,0003	0,0002
25	0,2395	0,2395	0,2393	0,2392	0,0003	0,0003	0,0001
26	0,2494	0,2495	0,2494	0,2494	0,0000	0,0001	0,0000
27	0,2590	0,2595	0,2593	0,2593	0,0003	0,0002	0,0000
28	0,2699	0,2690	0,2692	0,2693	0,0006	0,0003	0,0001
29	0,2790	0,2795	0,2793	0,2793	0,0003	0,0002	0,0000
30	0,2890	0,2891	0,2893	0,2894	0,0004	0,0003	0,0001
31	0,2996	0,2996	0,2994	0,2995	0,0001	0,0001	0,0001
32	0,3087	0,3092	0,3093	0,3093	0,0006	0,0001	0,0000
33	0,3191	0,3192	0,3193	0,3192	0,0001	0,0000	0,0001
34	0,3294	0,3292	0,3293	0,3293	0,0001	0,0001	0,0000
35	0,3387	0,3392	0,3392	0,3394	0,0007	0,0002	0,0002
36	0,3489	0,3492	0,3493	0,3493	0,0004	0,0001	0,0000
37	0,3587	0,3592	0,3592	0,3592	0,0005	0,0000	0,0000
38	0,3690	0,3692	0,3692	0,3692	0,0002	0,0000	0,0000
39	0,3796	0,3792	0,3791	0,3792	0,0004	0,0000	0,0001
40	0,3887	0,3893	0,3892	0,3892	0,0005	0,0001	0,0000
41	0,3994	0,3994	0,3993	0,3992	0,0002	0,0002	0,0001
42	0,4094	0,4094	0,4092	0,4092	0,0002	0,0002	0,0000
43	0,4190	0,4194	0,4192	0,4191	0,0001	0,0003	0,0001
44	0,4299	0,4294	0,4292	0,4291	0,0008	0,0003	0,0001
45	0,4388	0,4389	0,4391	0,4392	0,0004	0,0003	0,0001
46	0,4491	0,4489	0,4491	0,4492	0,0001	0,0003	0,0001
47	0,4598	0,4589	0,4591	0,4592	0,0006	0,0003	0,0001
48	0,4687	0,4690	0,4690	0,4692	0,0005	0,0002	0,0002
49	0,4798	0,4790	0,4791	0,4792	0,0006	0,0002	0,0001
50	0,4887	0,4890	0,4891	0,4893	0,0006	0,0003	0,0002
51	0,4992	0,4991	0,4992	0,4993	0,0001	0,0002	0,0001
					σ_E	σ_E	σ_E
					0,000270	0,000119	0,000106

Tabela 5-24: Efeito das ferramentas de medição em escala ordinal/racional sobre a exatidão do IMVC-E_3.

Definidas as ferramentas de medição em escala ordinal e racional, ou seja, o par *{transformada de Hough, método dos mínimos quadrados}*, investigou-se o efeito dos parâmetros da escala nominal. Adotando-se alguns valores arbitrários de limiar (159, 161, 165, 167) (ver tabela 5-25), nota-se que é possível obter medidas melhores do que as induzidas pelo valor (163) estimado pelo método de Otsu[8]. Todavia, em uma medição

[8] O impacto desse parâmetro, contudo, diminui substancialmente quando se esqueletoniza a imagem.

rotineira, o operador do IMVC-E_3 não teria critérios para selecionar um limiar mais adequado, o que torna a limiarização automática a única possível de fato.

			d_i			d_{REF}			e_i		
Limiar T	159	161	163	165	167		159	161	163	165	167
01	-0,0004	-0,0004	-0,0005	-0,0004	-0,0005	0,0000	0,0004	0,0004	0,0005	0,0004	0,0005
02	0,0093	0,0093	0,0093	0,0092	0,0092	0,0096	0,0003	0,0003	0,0003	0,0004	0,0004
03	0,0190	0,0191	0,0191	0,0191	0,0191	0,0192	0,0002	0,0001	0,0001	0,0001	0,0001
04	0,0290	0,0290	0,0290	0,0290	0,0290	0,0292	0,0002	0,0002	0,0002	0,0002	0,0002
05	0,0389	0,0389	0,0389	0,0389	0,0389	0,0392	0,0003	0,0003	0,0003	0,0003	0,0003
06	0,0489	0,0489	0,0489	0,0489	0,0489	0,0491	0,0002	0,0002	0,0002	0,0002	0,0002
07	0,0590	0,0590	0,0590	0,0590	0,0590	0,0591	0,0001	0,0001	0,0001	0,0001	0,0001
08	0,0690	0,0690	0,0690	0,0690	0,0690	0,0691	0,0001	0,0001	0,0001	0,0001	0,0001
09	0,0790	0,0790	0,0790	0,0790	0,0790	0,0790	0,0000	0,0000	0,0000	0,0000	0,0000
10	0,0891	0,0890	0,0891	0,0891	0,0891	0,0890	0,0001	0,0000	0,0001	0,0001	0,0001
11	0,0991	0,0991	0,0991	0,0991	0,0991	0,0991	0,0000	0,0000	0,0000	0,0000	0,0000
12	0,1091	0,1091	0,1091	0,1091	0,1091	0,1090	0,0001	0,0001	0,0001	0,0001	0,0001
13	0,1192	0,1192	0,1192	0,1192	0,1192	0,1190	0,0002	0,0002	0,0002	0,0002	0,0002
14	0,1293	0,1293	0,1293	0,1293	0,1293	0,1290	0,0003	0,0003	0,0003	0,0003	0,0003
15	0,1392	0,1392	0,1392	0,1392	0,1392	0,1390	0,0002	0,0002	0,0002	0,0002	0,0002
16	0,1492	0,1492	0,1492	0,1492	0,1492	0,1490	0,0002	0,0002	0,0002	0,0002	0,0002
17	0,1592	0,1592	0,1592	0,1592	0,1593	0,1589	0,0003	0,0003	0,0003	0,0003	0,0004
18	0,1692	0,1692	0,1692	0,1692	0,1692	0,1690	0,0002	0,0002	0,0002	0,0002	0,0002
19	0,1793	0,1793	0,1793	0,1793	0,1793	0,1790	0,0003	0,0003	0,0003	0,0003	0,0003
20	0,1893	0,1893	0,1893	0,1893	0,1893	0,1891	0,0002	0,0002	0,0002	0,0002	0,0002
21	0,1994	0,1994	0,1994	0,1994	0,1994	0,1992	0,0002	0,0002	0,0002	0,0002	0,0002
22	0,2093	0,2093	0,2093	0,2093	0,2093	0,2092	0,0001	0,0001	0,0001	0,0001	0,0001
23	0,2193	0,2193	0,2193	0,2193	0,2193	0,2192	0,0001	0,0001	0,0001	0,0001	0,0001
24	0,2294	0,2294	0,2294	0,2294	0,2294	0,2292	0,0002	0,0002	0,0002	0,0002	0,0002
25	0,2393	0,2393	0,2393	0,2393	0,2393	0,2392	0,0001	0,0001	0,0001	0,0001	0,0001
26	0,2494	0,2494	0,2494	0,2494	0,2494	0,2494	0,0000	0,0000	0,0000	0,0000	0,0000
27	0,2593	0,2593	0,2593	0,2593	0,2593	0,2593	0,0000	0,0000	0,0000	0,0000	0,0000
28	0,2692	0,2692	0,2692	0,2692	0,2692	0,2693	0,0001	0,0001	0,0001	0,0001	0,0001
29	0,2793	0,2793	0,2793	0,2793	0,2793	0,2793	0,0000	0,0000	0,0000	0,0000	0,0000
30	0,2893	0,2893	0,2893	0,2893	0,2893	0,2894	0,0001	0,0001	0,0001	0,0001	0,0001
31	0,2994	0,2994	0,2994	0,2994	0,2995	0,2995	0,0001	0,0001	0,0001	0,0001	0,0000
32	0,3094	0,3093	0,3093	0,3093	0,3093	0,3093	0,0001	0,0000	0,0000	0,0000	0,0000
33	0,3193	0,3193	0,3193	0,3193	0,3193	0,3192	0,0001	0,0001	0,0001	0,0001	0,0001
34	0,3293	0,3293	0,3293	0,3293	0,3293	0,3293	0,0000	0,0000	0,0000	0,0000	0,0000
35	0,3392	0,3392	0,3392	0,3392	0,3392	0,3394	0,0002	0,0002	0,0002	0,0002	0,0002
36	0,3493	0,3493	0,3493	0,3493	0,3493	0,3493	0,0000	0,0000	0,0000	0,0000	0,0000
37	0,3592	0,3592	0,3592	0,3592	0,3592	0,3592	0,0000	0,0000	0,0000	0,0000	0,0000
38	0,3691	0,3691	0,3692	0,3691	0,3692	0,3692	0,0001	0,0001	0,0000	0,0001	0,0000
39	0,3791	0,3791	0,3791	0,3791	0,3791	0,3792	0,0001	0,0001	0,0001	0,0001	0,0001
40	0,3892	0,3892	0,3892	0,3892	0,3892	0,3892	0,0000	0,0000	0,0000	0,0000	0,0000
41	0,3993	0,3993	0,3993	0,3993	0,3993	0,3992	0,0001	0,0001	0,0001	0,0001	0,0001
42	0,4092	0,4092	0,4092	0,4092	0,4092	0,4092	0,0000	0,0000	0,0000	0,0000	0,0000
43	0,4192	0,4192	0,4192	0,4192	0,4192	0,4191	0,0001	0,0001	0,0001	0,0001	0,0001
44	0,4292	0,4292	0,4292	0,4292	0,4292	0,4291	0,0001	0,0001	0,0001	0,0001	0,0001
45	0,4391	0,4391	0,4391	0,4391	0,4391	0,4392	0,0001	0,0001	0,0001	0,0001	0,0001
46	0,4491	0,4491	0,4491	0,4490	0,4491	0,4492	0,0001	0,0001	0,0001	0,0002	0,0001
47	0,4591	0,4591	0,4591	0,4591	0,4591	0,4592	0,0001	0,0001	0,0001	0,0001	0,0001
48	0,4690	0,4690	0,4690	0,4690	0,4690	0,4692	0,0002	0,0002	0,0002	0,0002	0,0002
49	0,4792	0,4792	0,4791	0,4791	0,4791	0,4792	0,0000	0,0000	0,0001	0,0001	0,0001
50	0,4891	0,4891	0,4891	0,4891	0,4891	0,4893	0,0002	0,0002	0,0002	0,0002	0,0002
51	0,4992	0,4992	0,4992	0,4992	0,4992	0,4993	0,0001	0,0001	0,0001	0,0001	0,0001
							σ_E	σ_E	σ_E	σ_E	σ_E
							0,000099	0,000101	0,000106	0,000103	0,000115

Tabela 5-25: Efeito do valor do limiar sobre a exatidão do IMVC-E_3.

Por outro lado, a esqueletonização da imagem binarizada melhora nitidamente a qualidade dos resultados, como mostram os dados da tabela 5-26, onde se utilizaram as 5 imagens da série 88,5° para medir o desempenho do IMVC-E_3 em sua condição ideal de operação, qual seja: escala nominal: *limiarização* = automática; *esqueletonização* =

ausente; escala ordinal: *transformada de Hough* ($d\theta$=0,50; s_ρ=6; L=150; E=1); escala racional: *ajuste por mínimos quadrados*. Como se pode notar, nessas condições, a repetibilidade média desse instrumento já é 5 vezes superior à do ensaio convencional !

T	d_1	d_2	d_3	d_4	d_5	d	s	δ	d_{REF}	e
01	-0,0004	-0,0004	-0,0003	-0,0004	-0,0004	-0,00038	0,000045	0,000056	0,0000	0,0004
02	0,0093	0,0093	0,0093	0,0093	0,0093	0,00930	0,000000	0,000000	0,0096	0,0003
03	0,0190	0,0191	0,0191	0,0191	0,0190	0,01906	0,000055	0,000068	0,0192	0,0001
04	0,0290	0,0291	0,0289	0,0290	0,0290	0,02900	0,000071	0,000088	0,0292	0,0002
05	0,0390	0,0389	0,0389	0,0389	0,0390	0,03894	0,000055	0,000068	0,0392	0,0003
06	0,0489	0,0489	0,0490	0,0489	0,0490	0,04894	0,000055	0,000068	0,0491	0,0002
07	0,0590	0,0590	0,0591	0,0589	0,0591	0,05902	0,000084	0,000104	0,0591	0,0001
08	0,0690	0,0690	0,0690	0,0691	0,0690	0,06902	0,000045	0,000056	0,0691	0,0001
09	0,0790	0,0790	0,0791	0,0791	0,0791	0,07906	0,000055	0,000068	0,0790	0,0001
10	0,0890	0,0891	0,0891	0,0890	0,0891	0,08906	0,000055	0,000068	0,0890	0,0001
11	0,0992	0,0992	0,0991	0,0991	0,0991	0,09914	0,000055	0,000068	0,0991	0,0000
12	0,1091	0,1091	0,1091	0,1091	0,1091	0,10910	0,000000	0,000000	0,1090	0,0001
13	0,1191	0,1191	0,1191	0,1191	0,1191	0,11910	0,000000	0,000000	0,1190	0,0001
14	0,1292	0,1292	0,1292	0,1291	0,1291	0,12916	0,000055	0,000068	0,1290	0,0002
15	0,1392	0,1392	0,1391	0,1392	0,1391	0,13916	0,000055	0,000068	0,1390	0,0002
16	0,1493	0,1492	0,1491	0,1493	0,1492	0,14922	0,000084	0,000104	0,1490	0,0002
17	0,1592	0,1592	0,1592	0,1592	0,1592	0,15920	0,000000	0,000000	0,1589	0,0003
18	0,1692	0,1692	0,1692	0,1692	0,1692	0,16920	0,000000	0,000000	0,1690	0,0002
19	0,1793	0,1793	0,1793	0,1793	0,1793	0,17930	0,000000	0,000000	0,1790	0,0003
20	0,1893	0,1893	0,1893	0,1894	0,1893	0,18932	0,000045	0,000056	0,1891	0,0002
21	0,1994	0,1994	0,1994	0,1994	0,1994	0,19940	0,000000	0,000000	0,1992	0,0002
22	0,2093	0,2094	0,2093	0,2093	0,2093	0,20932	0,000045	0,000056	0,2092	0,0001
23	0,2193	0,2194	0,2193	0,2193	0,2193	0,21932	0,000045	0,000056	0,2192	0,0001
24	0,2294	0,2294	0,2294	0,2294	0,2293	0,22938	0,000045	0,000056	0,2292	0,0002
25	0,2393	0,2393	0,2392	0,2393	0,2393	0,23928	0,000045	0,000056	0,2392	0,0001
26	0,2493	0,2493	0,2494	0,2494	0,2494	0,24936	0,000055	0,000068	0,2494	0,0000
27	0,2594	0,2593	0,2594	0,2595	0,2594	0,25940	0,000071	0,000088	0,2593	0,0001
28	0,2693	0,2692	0,2693	0,2694	0,2693	0,26930	0,000071	0,000088	0,2693	0,0000
29	0,2793	0,2794	0,2795	0,2793	0,2794	0,27938	0,000084	0,000104	0,2793	0,0001
30	0,2893	0,2893	0,2894	0,2893	0,2893	0,28932	0,000045	0,000056	0,2894	0,0001
31	0,2994	0,2994	0,2994	0,2994	0,2994	0,29940	0,000000	0,000000	0,2995	0,0001
32	0,3093	0,3093	0,3093	0,3093	0,3094	0,30932	0,000045	0,000056	0,3093	0,0000
33	0,3193	0,3193	0,3193	0,3193	0,3192	0,31928	0,000045	0,000056	0,3192	0,0001
34	0,3293	0,3293	0,3293	0,3293	0,3293	0,32930	0,000000	0,000000	0,3293	0,0000
35	0,3393	0,3392	0,3393	0,3393	0,3392	0,33926	0,000055	0,000068	0,3394	0,0001
36	0,3493	0,3492	0,3492	0,3493	0,3492	0,34924	0,000055	0,000068	0,3493	0,0001
37	0,3592	0,3592	0,3592	0,3592	0,3592	0,35920	0,000000	0,000000	0,3592	0,0000
38	0,3691	0,3691	0,3691	0,3690	0,3690	0,36906	0,000055	0,000068	0,3692	0,0001
39	0,3791	0,3792	0,3791	0,3791	0,3791	0,37912	0,000045	0,000056	0,3792	0,0001
40	0,3892	0,3891	0,3892	0,3892	0,3892	0,38918	0,000045	0,000056	0,3892	0,0000
41	0,3993	0,3993	0,3993	0,3993	0,3993	0,39930	0,000000	0,000000	0,3992	0,0001
42	0,4092	0,4092	0,4092	0,4092	0,4092	0,40920	0,000000	0,000000	0,4092	0,0000
43	0,4192	0,4192	0,4192	0,4192	0,4192	0,41920	0,000000	0,000000	0,4191	0,0001
44	0,4292	0,4291	0,4291	0,4292	0,4292	0,42916	0,000055	0,000068	0,4291	0,0001
45	0,4391	0,4391	0,4391	0,4390	0,4391	0,43908	0,000045	0,000056	0,4392	0,0001
46	0,4491	0,4491	0,4491	0,4491	0,4491	0,44910	0,000000	0,000000	0,4492	0,0001
47	0,4591	0,4591	0,4591	0,4591	0,4591	0,45910	0,000000	0,000000	0,4592	0,0001
48	0,4691	0,4690	0,4691	0,4691	0,4691	0,46908	0,000045	0,000056	0,4692	0,0001
49	0,4792	0,4792	0,4791	0,4791	0,4791	0,47914	0,000055	0,000068	0,4792	0,0001
50	0,4891	0,4890	0,4891	0,4891	0,4892	0,48910	0,000071	0,000088	0,4893	0,0002
51	0,4992	0,4993	0,4992	0,4992	0,4992	0,49922	0,000045	0,000056	0,4993	0,0001
							σ_R			σ_E
							0,000039			0,000085

Tabela 5-26: Medição do desempenho do IMVC-E_3 *em sua condição ótima de operação.*

5.5.4 Medição do Desempenho do IMVC-E_4

Adotando para as escalas ordinal e racional os parâmetros ótimos determinados para o IMVC-E_3, investigou-se a influência dos demais parâmetros da escala nominal sobre o desempenho do IMVC-E_4, utilizando-se imagens da série 88,5°.

Analisando-se inicialmente a largura w do filtro de suavização, observou-se que, independentemente do tipo de detector utilizado, o valor w=3 era o que correspondia ao limiar que melhor segmentava as imagens de bordas, sendo que: 1º) valores menores preservavam ruído; 2º) valores maiores eliminavam partes dos objetos relevantes da cena. Esses efeitos são observados na figura 5.9, onde a imagem de bordas resultante da aplicação do operador de Roberts é limiarizada segundo os valores de limiar $L(w=0)$, $L(w=3)$ e $L(w=5)$, onde L é calculado a partir das expressões 3-10 e 3-6.

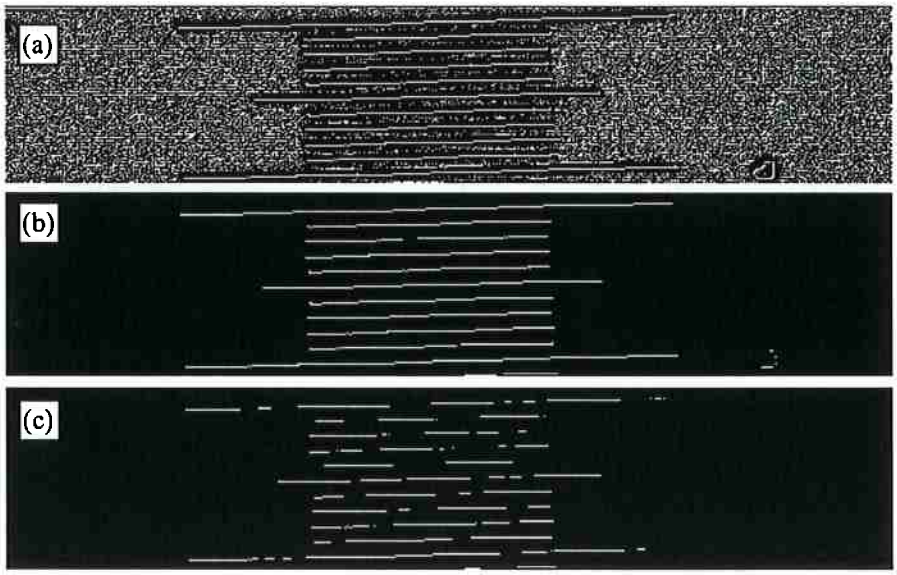

Figura 5.9: Influência da largura do filtro de suavização: (a) w=0; (b) w=3; (c) w=5.

Para avaliar se a particular borda — claro→escura, escuro→clara ou medial — tomada como referência para as estimações das posições dos traços da imagem, exercia influência sobre a exatidão e repetibilidade das medidas, realizaram-se experimentos com as 5 imagens da série adotando-se os seguintes parâmetros para a escala nominal:

largura do filtro de suavização = 3, *tipo de detector* = Sobel, *tipo de Borda* ∈{claro→ escura, escuro→clara, medial}, obtendo-se, para cada caso, os valores médios d_i das posições dos traços, seus desvios e_i em relação à referência e as estimativas de exatidão e repetibilidade do IMVC-E_4. Esses dados são apresentados nas tabelas 5-27 a 5-29, onde se pode observar o substancial aumento da repetibilidade do instrumento quando se utilizam as bordas mediais dos traços.

T	d_1	d_2	d_3	d_4	d_5	d	s	d_{REF}	$d-d_{REF}$
01	-0,0004	-0,0004	-0,0004	-0,0004	-0,0004	-0,00040	0,00000	0,0000	0,0004
02	0,0093	0,0092	0,0093	0,0092	0,0093	0,00926	0,00005	0,0096	0,0003
03	0,0191	0,0191	0,0190	0,0191	0,0191	0,01908	0,00004	0,0192	0,0001
04	0,0290	0,0290	0,0290	0,0290	0,0290	0,02900	0,00000	0,0292	0,0002
05	0,0389	0,0389	0,0389	0,0389	0,0390	0,03892	0,00004	0,0392	0,0003
06	0,0490	0,0490	0,0490	0,0489	0,0489	0,04896	0,00005	0,0491	0,0001
07	0,0590	0,0591	0,0590	0,0591	0,0590	0,05904	0,00005	0,0591	0,0001
08	0,0691	0,0690	0,0690	0,0690	0,0690	0,06902	0,00004	0,0691	0,0001
09	0,0791	0,0791	0,0791	0,0791	0,0791	0,07910	0,00000	0,0790	0,0001
10	0,0890	0,0891	0,0891	0,0891	0,0890	0,08906	0,00005	0,0890	0,0001
11	0,0991	0,0991	0,0991	0,0992	0,0991	0,09912	0,00004	0,0991	0,0000
12	0,1092	0,1091	0,1091	0,1091	0,1091	0,10912	0,00004	0,1090	0,0001
13	0,1191	0,1191	0,1191	0,1191	0,1191	0,11910	0,00000	0,1190	0,0001
14	0,1292	0,1292	0,1292	0,1292	0,1292	0,12920	0,00000	0,1290	0,0002
15	0,1392	0,1391	0,1391	0,1392	0,1392	0,13916	0,00005	0,1390	0,0002
16	0,1492	0,1492	0,1493	0,1492	0,1493	0,14924	0,00005	0,1490	0,0002
17	0,1592	0,1593	0,1593	0,1592	0,1592	0,15924	0,00005	0,1589	0,0003
18	0,1692	0,1692	0,1692	0,1692	0,1692	0,16920	0,00000	0,1690	0,0002
19	0,1793	0,1792	0,1792	0,1793	0,1793	0,17926	0,00005	0,1790	0,0003
20	0,1893	0,1893	0,1893	0,1893	0,1893	0,18930	0,00000	0,1891	0,0002
21	0,1994	0,1994	0,1994	0,1994	0,1994	0,19940	0,00000	0,1992	0,0002
22	0,2093	0,2093	0,2093	0,2094	0,2093	0,20932	0,00004	0,2092	0,0001
23	0,2194	0,2193	0,2193	0,2193	0,2193	0,21932	0,00004	0,2192	0,0001
24	0,2293	0,2294	0,2295	0,2293	0,2294	0,22938	0,00008	0,2292	0,0002
25	0,2393	0,2393	0,2393	0,2393	0,2392	0,23928	0,00004	0,2392	0,0001
26	0,2494	0,2493	0,2494	0,2494	0,2493	0,24936	0,00005	0,2494	0,0000
27	0,2594	0,2593	0,2593	0,2593	0,2594	0,25934	0,00005	0,2593	0,0000
28	0,2693	0,2693	0,2693	0,2693	0,2693	0,26930	0,00000	0,2693	0,0000
29	0,2794	0,2794	0,2794	0,2794	0,2793	0,27938	0,00004	0,2793	0,0001
30	0,2893	0,2893	0,2893	0,2894	0,2893	0,28932	0,00004	0,2894	0,0001
31	0,2995	0,2994	0,2994	0,2994	0,2995	0,29944	0,00005	0,2995	0,0001
32	0,3093	0,3094	0,3094	0,3093	0,3093	0,30934	0,00005	0,3093	0,0000
33	0,3192	0,3193	0,3193	0,3193	0,3192	0,31926	0,00005	0,3192	0,0001
34	0,3293	0,3293	0,3293	0,3293	0,3293	0,32930	0,00000	0,3293	0,0000
35	0,3393	0,3392	0,3392	0,3393	0,3393	0,33926	0,00005	0,3394	0,0001
36	0,3493	0,3493	0,3493	0,3492	0,3493	0,34928	0,00004	0,3493	0,0000
37	0,3592	0,3593	0,3593	0,3592	0,3592	0,35924	0,00005	0,3592	0,0000
38	0,3691	0,3691	0,3691	0,3691	0,3691	0,36910	0,00000	0,3692	0,0001
39	0,3792	0,3791	0,3791	0,3792	0,3791	0,37914	0,00005	0,3792	0,0001
40	0,3892	0,3891	0,3892	0,3891	0,3892	0,38916	0,00005	0,3892	0,0000
41	0,3993	0,3993	0,3993	0,3993	0,3993	0,39930	0,00000	0,3992	0,0001
42	0,4092	0,4092	0,4092	0,4092	0,4092	0,40920	0,00000	0,4092	0,0000
43	0,4192	0,4192	0,4192	0,4192	0,4192	0,41920	0,00000	0,4191	0,0001
44	0,4292	0,4292	0,4292	0,4291	0,4292	0,42918	0,00004	0,4291	0,0001
45	0,4390	0,4390	0,4391	0,4391	0,4391	0,43906	0,00005	0,4392	0,0001
46	0,4491	0,4491	0,4491	0,4492	0,4491	0,44912	0,00004	0,4492	0,0001
47	0,4591	0,4591	0,4591	0,4591	0,4591	0,45910	0,00000	0,4592	0,0001
48	0,4691	0,4690	0,4690	0,4690	0,4691	0,46904	0,00005	0,4692	0,0002
49	0,4791	0,4792	0,4792	0,4791	0,4791	0,47914	0,00005	0,4792	0,0001
50	0,4890	0,4891	0,4891	0,4891	0,4890	0,48906	0,00005	0,4893	0,0002
51	0,4992	0,4991	0,4992	0,4992	0,4992	0,49918	0,00004	0,4993	0,0001
							σ_R		σ_E
							0,000036		0,000091

Tabela 5-27: Repetibilidade e exatidão do IMVC-E_4, utilizando-se bordas claro→escuras.

T	d_1	d_2	d_3	d_4	d_5	d	s	d_{REF}	$d-d_{REF}$
01	-0,0004	-0,0004	-0,0004	-0,0004	-0,0003	-0,00038	0,00004	0,0000	0,00038
02	0,0093	0,0093	0,0093	0,0093	0,0093	0,00930	0,00000	0,0096	0,00030
03	0,0191	0,0191	0,0191	0,0191	0,0191	0,01910	0,00000	0,0192	0,00010
04	0,0290	0,0290	0,0290	0,0290	0,0290	0,02900	0,00000	0,0292	0,00020
05	0,0389	0,0390	0,0390	0,0390	0,0389	0,03896	0,00005	0,0392	0,00024
06	0,0490	0,0490	0,0490	0,0490	0,0490	0,04900	0,00000	0,0491	0,00010
07	0,0591	0,0590	0,0590	0,0591	0,0590	0,05904	0,00005	0,0591	0,00006
08	0,0690	0,0690	0,0690	0,0690	0,0690	0,06900	0,00000	0,0691	0,00010
09	0,0791	0,0791	0,0791	0,0790	0,0790	0,07906	0,00005	0,0790	0,00006
10	0,0890	0,0890	0,0890	0,0891	0,0890	0,08902	0,00004	0,0890	0,00002
11	0,0992	0,0991	0,0991	0,0991	0,0991	0,09912	0,00004	0,0991	0,00002
12	0,1090	0,1090	0,1091	0,1091	0,1091	0,10906	0,00005	0,1090	0,00006
13	0,1191	0,1191	0,1191	0,1191	0,1191	0,11910	0,00000	0,1190	0,00010
14	0,1292	0,1291	0,1292	0,1292	0,1292	0,12918	0,00004	0,1290	0,00018
15	0,1392	0,1392	0,1391	0,1391	0,1392	0,13916	0,00005	0,1390	0,00016
16	0,1492	0,1493	0,1492	0,1492	0,1492	0,14922	0,00004	0,1490	0,00022
17	0,1592	0,1592	0,1592	0,1592	0,1592	0,15920	0,00000	0,1589	0,00030
18	0,1692	0,1692	0,1692	0,1692	0,1691	0,16918	0,00004	0,1690	0,00018
19	0,1793	0,1792	0,1793	0,1793	0,1793	0,17928	0,00004	0,1790	0,00028
20	0,1893	0,1893	0,1893	0,1893	0,1893	0,18930	0,00000	0,1891	0,00020
21	0,1994	0,1994	0,1994	0,1994	0,1994	0,19940	0,00000	0,1992	0,00020
22	0,2094	0,2093	0,2093	0,2093	0,2094	0,20934	0,00005	0,2092	0,00014
23	0,2193	0,2193	0,2193	0,2193	0,2193	0,21930	0,00000	0,2192	0,00010
24	0,2294	0,2294	0,2294	0,2293	0,2294	0,22938	0,00004	0,2292	0,00018
25	0,2392	0,2393	0,2393	0,2393	0,2393	0,23928	0,00004	0,2392	0,00008
26	0,2494	0,2493	0,2493	0,2494	0,2494	0,24936	0,00005	0,2494	0,00004
27	0,2593	0,2594	0,2594	0,2593	0,2593	0,25934	0,00005	0,2593	0,00004
28	0,2693	0,2693	0,2693	0,2692	0,2693	0,26928	0,00004	0,2693	0,00002
29	0,2794	0,2793	0,2793	0,2794	0,2794	0,27936	0,00005	0,2793	0,00006
30	0,2894	0,2893	0,2893	0,2893	0,2893	0,28932	0,00004	0,2894	0,00008
31	0,2994	0,2994	0,2995	0,2994	0,2995	0,29944	0,00005	0,2995	0,00006
32	0,3093	0,3093	0,3093	0,3093	0,3093	0,30930	0,00000	0,3093	0,00000
33	0,3193	0,3192	0,3192	0,3193	0,3193	0,31926	0,00005	0,3192	0,00006
34	0,3293	0,3292	0,3293	0,3293	0,3293	0,32928	0,00004	0,3293	0,00002
35	0,3393	0,3393	0,3393	0,3392	0,3392	0,33926	0,00005	0,3394	0,00014
36	0,3492	0,3493	0,3493	0,3492	0,3492	0,34924	0,00005	0,3493	0,00006
37	0,3592	0,3592	0,3592	0,3592	0,3591	0,35918	0,00004	0,3592	0,00002
38	0,3691	0,3691	0,3691	0,3691	0,3691	0,36910	0,00000	0,3692	0,00010
39	0,3791	0,3791	0,3791	0,3791	0,3792	0,37912	0,00004	0,3792	0,00008
40	0,3891	0,3892	0,3892	0,3891	0,3891	0,38914	0,00005	0,3892	0,00006
41	0,3993	0,3993	0,3993	0,3993	0,3993	0,39930	0,00000	0,3992	0,00010
42	0,4092	0,4092	0,4092	0,4093	0,4092	0,40922	0,00004	0,4092	0,00002
43	0,4192	0,4192	0,4192	0,4192	0,4192	0,41920	0,00000	0,4191	0,00010
44	0,4292	0,4292	0,4292	0,4292	0,4292	0,42920	0,00000	0,4291	0,00010
45	0,4391	0,4391	0,4391	0,4390	0,4391	0,43908	0,00004	0,4392	0,00012
46	0,4492	0,4491	0,4491	0,4491	0,4491	0,44912	0,00004	0,4492	0,00008
47	0,4591	0,4591	0,4591	0,4591	0,4591	0,45910	0,00000	0,4592	0,00010
48	0,4691	0,4691	0,4691	0,4691	0,4691	0,46910	0,00000	0,4692	0,00010
49	0,4791	0,4792	0,4791	0,4791	0,4792	0,47914	0,00005	0,4792	0,00006
50	0,4891	0,4891	0,4891	0,4891	0,4890	0,48908	0,00004	0,4893	0,00022
51	0,4992	0,4991	0,4992	0,4992	0,4992	0,49918	0,00004	0,4993	0,00012
							σ_R		σ_E
							0,000033		0,000084

Tabela 5-28: Repetibilidade e exatidão do IMVC-E_4, utilizando-se bordas escuro→claras.

T	d_1	d_2	d_3	d_4	d_5	d	s	d_{REF}	$d-d_{REF}$
01	-0,0004	-0,0004	-0,0004	-0,0004	-0,0004	-0,00040	0,00000	0,0000	0,0004
02	0,0093	0,0093	0,0093	0,0093	0,0093	0,00930	0,00000	0,0096	0,0003
03	0,0191	0,0191	0,0191	0,0191	0,0191	0,01910	0,00000	0,0192	0,0001
04	0,0290	0,0290	0,0290	0,0290	0,0290	0,02900	0,00000	0,0292	0,0002
05	0,0389	0,0390	0,0390	0,0389	0,0389	0,03894	0,00005	0,0392	0,0003
06	0,0490	0,0490	0,0490	0,0489	0,0489	0,04896	0,00005	0,0491	0,0001
07	0,0590	0,0590	0,0590	0,0591	0,0590	0,05902	0,00004	0,0591	0,0001
08	0,0691	0,0690	0,0690	0,0690	0,0690	0,06902	0,00004	0,0691	0,0001
09	0,0791	0,0791	0,0791	0,0791	0,0791	0,07910	0,00000	0,0790	0,0001
10	0,0890	0,0891	0,0891	0,0891	0,0890	0,08906	0,00005	0,0890	0,0001
11	0,0991	0,0991	0,0991	0,0991	0,0991	0,09910	0,00000	0,0991	0,0000
12	0,1091	0,1090	0,1091	0,1091	0,1091	0,10908	0,00004	0,1090	0,0001
13	0,1191	0,1191	0,1191	0,1191	0,1191	0,11910	0,00000	0,1190	0,0001
14	0,1292	0,1292	0,1292	0,1292	0,1292	0,12920	0,00000	0,1290	0,0002
15	0,1392	0,1391	0,1391	0,1392	0,1392	0,13916	0,00005	0,1390	0,0002
16	0,1492	0,1492	0,1493	0,1492	0,1492	0,14922	0,00004	0,1490	0,0002
17	0,1592	0,1593	0,1592	0,1592	0,1592	0,15922	0,00004	0,1589	0,0003
18	0,1692	0,1692	0,1692	0,1692	0,1692	0,16920	0,00000	0,1690	0,0002
19	0,1793	0,1792	0,1792	0,1793	0,1793	0,17926	0,00005	0,1790	0,0003
20	0,1893	0,1893	0,1893	0,1893	0,1893	0,18930	0,00000	0,1891	0,0002
21	0,1994	0,1994	0,1994	0,1994	0,1994	0,19940	0,00000	0,1992	0,0002
22	0,2093	0,2093	0,2093	0,2094	0,2093	0,20932	0,00004	0,2092	0,0001
23	0,2193	0,2193	0,2193	0,2193	0,2193	0,21930	0,00000	0,2192	0,0001
24	0,2294	0,2294	0,2294	0,2293	0,2294	0,22938	0,00004	0,2292	0,0002
25	0,2393	0,2393	0,2393	0,2393	0,2393	0,23930	0,00000	0,2392	0,0001
26	0,2494	0,2493	0,2493	0,2494	0,2494	0,24936	0,00005	0,2494	0,0000
27	0,2594	0,2594	0,2593	0,2593	0,2594	0,25936	0,00005	0,2593	0,0001
28	0,2693	0,2693	0,2693	0,2693	0,2693	0,26930	0,00000	0,2693	0,0000
29	0,2794	0,2793	0,2793	0,2794	0,2794	0,27936	0,00005	0,2793	0,0001
30	0,2893	0,2893	0,2893	0,2893	0,2893	0,28930	0,00000	0,2894	0,0001
31	0,2994	0,2994	0,2995	0,2994	0,2995	0,29944	0,00005	0,2995	0,0001
32	0,3093	0,3093	0,3093	0,3093	0,3093	0,30930	0,00000	0,3093	0,0000
33	0,3192	0,3193	0,3193	0,3193	0,3192	0,31926	0,00005	0,3192	0,0001
34	0,3293	0,3293	0,3293	0,3293	0,3293	0,32930	0,00000	0,3293	0,0000
35	0,3393	0,3392	0,3392	0,3392	0,3392	0,33922	0,00004	0,3394	0,0002
36	0,3492	0,3493	0,3493	0,3492	0,3492	0,34924	0,00005	0,3493	0,0001
37	0,3592	0,3592	0,3592	0,3592	0,3592	0,35920	0,00000	0,3592	0,0000
38	0,3691	0,3691	0,3691	0,3691	0,3691	0,36910	0,00000	0,3692	0,0001
39	0,3792	0,3791	0,3791	0,3791	0,3792	0,37914	0,00005	0,3792	0,0001
40	0,3891	0,3892	0,3892	0,3891	0,3892	0,38916	0,00005	0,3892	0,0000
41	0,3993	0,3993	0,3993	0,3993	0,3993	0,39930	0,00000	0,3992	0,0001
42	0,4092	0,4092	0,4092	0,4092	0,4092	0,40920	0,00000	0,4092	0,0000
43	0,4192	0,4192	0,4192	0,4192	0,4192	0,41920	0,00000	0,4191	0,0001
44	0,4292	0,4292	0,4292	0,4292	0,4292	0,42920	0,00000	0,4291	0,0001
45	0,4391	0,4391	0,4391	0,4391	0,4391	0,43910	0,00000	0,4392	0,0001
46	0,4491	0,4491	0,4491	0,4491	0,4491	0,44910	0,00000	0,4492	0,0001
47	0,4591	0,4591	0,4591	0,4591	0,4591	0,45910	0,00000	0,4592	0,0001
48	0,4691	0,4691	0,4691	0,4691	0,4691	0,46910	0,00000	0,4692	0,0001
49	0,4791	0,4792	0,4792	0,4791	0,4791	0,47914	0,00005	0,4792	0,0001
50	0,4890	0,4891	0,4891	0,4891	0,4890	0,48906	0,00005	0,4893	0,0002
51	0,4992	0,4991	0,4992	0,4992	0,4992	0,49918	0,00004	0,4993	0,0001
							σ_R		σ_E
							0,000024		0,000089

Tabela 5-29: Repetibilidade e exatidão do IMVC-E_4, utilizando-se bordas mediais.

É importante frisar que a capacidade de o IMVC-E_4 realizar medidas adotando uma das duas bordas claro→escuras ou escuro→claras da imagem, pode ser utilizada proveitosamente para se calcular um outro índice de qualidade da microescala baseado na incerteza do valor médio da espessura de seus traços. Conforme se apresenta na tabela 5-30 e no gráfico 5-10, medições realizadas com as 5 imagens da série

identificam, com 95% de confiança, uma incerteza δ_E na espessura E dos traços da microescala da ordem de 0,00005 μm.

T	E_1	E_2	E_3	E_4	E_5	E
01	0,00208	0,00205	0,00204	0,00207	0,00211	0,00207
02	0,00202	0,00202	0,00203	0,00210	0,00211	0,00205
03	0,00205	0,00205	0,00197	0,00201	0,00209	0,00203
04	0,00203	0,00197	0,00205	0,00204	0,00216	0,00205
05	0,00196	0,00194	0,00208	0,00202	0,00204	0,00201
06	0,00199	0,00197	0,00205	0,00201	0,00207	0,00202
07	0,00198	0,00196	0,00200	0,00203	0,00210	0,00201
08	0,00197	0,00207	0,00206	0,00199	0,00207	0,00203
09	0,00199	0,00203	0,00198	0,00196	0,00201	0,00200
10	0,00198	0,00193	0,00189	0,00200	0,00199	0,00196
11	0,00204	0,00201	0,00198	0,00203	0,00209	0,00203
12	0,00203	0,00199	0,00202	0,00196	0,00202	0,00200
13	0,00203	0,00193	0,00196	0,00204	0,00201	0,00199
14	0,00201	0,00194	0,00192	0,00205	0,00210	0,00200
15	0,00195	0,00191	0,00201	0,00197	0,00208	0,00199
16	0,00193	0,00200	0,00201	0,00194	0,00205	0,00199
17	0,00194	0,00193	0,00199	0,00201	0,00207	0,00199
18	0,00198	0,00193	0,00195	0,00202	0,00202	0,00198
19	0,00195	0,00195	0,00191	0,00198	0,00207	0,00197
20	0,00197	0,00200	0,00198	0,00187	0,00196	0,00195
21	0,00202	0,00195	0,00191	0,00199	0,00203	0,00198
22	0,00203	0,00198	0,00186	0,00198	0,00204	0,00198
23	0,00199	0,00192	0,00200	0,00195	0,00210	0,00199
24	0,00202	0,00196	0,00192	0,00199	0,00204	0,00199
25	0,00201	0,00198	0,00188	0,00202	0,00204	0,00199
26	0,00203	0,00193	0,00191	0,00199	0,00208	0,00199
27	0,00186	0,00198	0,00200	0,00197	0,00203	0,00197
28	0,00196	0,00197	0,00191	0,00188	0,00196	0,00194
29	0,00200	0,00190	0,00188	0,00193	0,00198	0,00193
30	0,00206	0,00202	0,00184	0,00200	0,00204	0,00199
31	0,00197	0,00201	0,00196	0,00197	0,00204	0,00199
32	0,00201	0,00194	0,00193	0,00203	0,00202	0,00198
33	0,00195	0,00184	0,00193	0,00201	0,00213	0,00197
34	0,00202	0,00193	0,00203	0,00202	0,00214	0,00203
35	0,00197	0,00201	0,00199	0,00194	0,00202	0,00199
36	0,00201	0,00197	0,00194	0,00203	0,00202	0,00199
37	0,00206	0,00197	0,00193	0,00200	0,00205	0,00200
38	0,00195	0,00186	0,00194	0,00210	0,00212	0,00199
39	0,00198	0,00190	0,00200	0,00200	0,00209	0,00199
40	0,00195	0,00193	0,00200	0,00201	0,00200	0,00198
41	0,00199	0,00197	0,00195	0,00206	0,00214	0,00202
42	0,00204	0,00194	0,00200	0,00201	0,00207	0,00201
43	0,00205	0,00208	0,00195	0,00202	0,00209	0,00204
44	0,00200	0,00199	0,00195	0,00199	0,00206	0,00200
45	0,00207	0,00196	0,00192	0,00206	0,00208	0,00202
46	0,00203	0,00193	0,00195	0,00201	0,00209	0,00200
47	0,00203	0,00190	0,00195	0,00205	0,00211	0,00201
48	0,00188	0,00196	0,00200	0,00205	0,00211	0,00200
49	0,00199	0,00195	0,00193	0,00204	0,00207	0,00200
50	0,00210	0,00195	0,00189	0,00203	0,00202	0,00200
51	0,00205	0,00196	0,00201	0,00200	0,00212	0,00203
						δ_E
						0,00005

Tabela 5-30: Incertezas das espessuras dos traços da microescala.

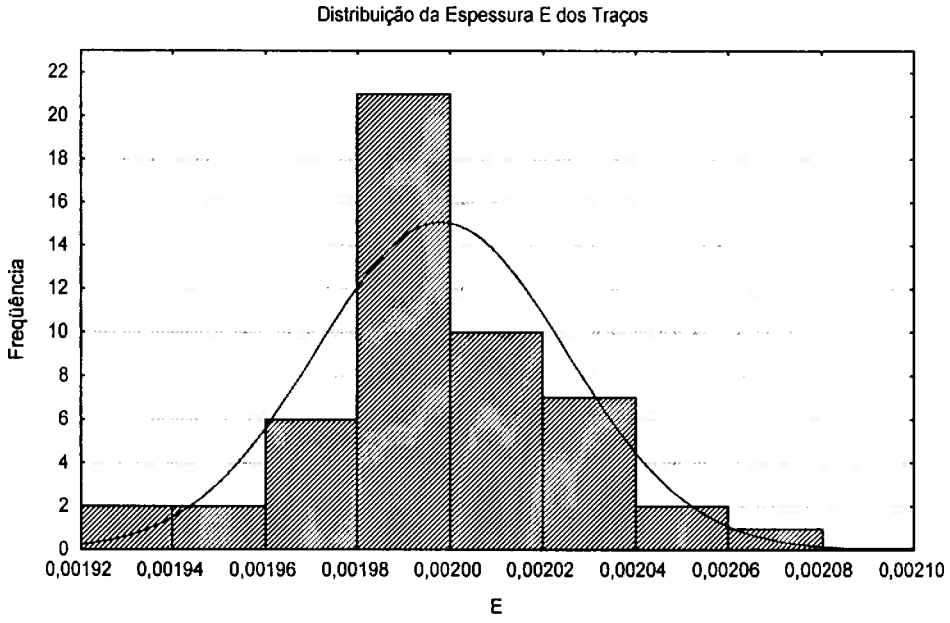

Figura 5.10: Distribuição da espessura dos traços da microescala.

O efeito do particular detector de bordas sobre a exatidão e repetibilidade das medidas foi investigado a partir de experimentos realizados com as 5 imagens da série, adotando-se os seguintes parâmetros para a escala nominal: *largura do filtro de suavização* = 3, *tipo de borda* = medial, *tipo de detector* $\in$ {Roberts, Sobel, Jensen, Lyvers}. Os resultados desses experimentos são apresentados na tabela 5-31, onde se observa o superior desempenho dos detectores de Roberts e Sobel tanto em relação a exatidão quanto em relação a repetibilidade.

O bom desempenho do detector de Roberts já era previsível, uma vez que, por causa da fina espessura dos traços da imagem, da ordem de 2 pixels, a máscara 2x2 é a que melhor adere ao modelo borda-degrau. Comparativamente, o detector de Sobel, apesar de utilizar uma máscara 3x3, compensa esse efeito negativo com a pré-filtragem de ruídos da imagem. Os maus resultados obtidos com o detector de Lyvers *et al* podem ser atribuídos ao tamanho da máscara utilizada (5x5), porém, a mesma explicação não justifica o baixo desempenho do detector de Jensen *et* Anastassiou, baseado em máscara

3x3 (mesma largura do detector de Sobel); restam, portanto, duas hipóteses para esse caso: erro de implementação ou possível viés no modelo do detector.

	Roberts			Sobel			Jensen			Lyvers		
T	d	$d-d_{REF}$	s	d	$d-d_{REF}$	s	d	$d-d_{REF}$	s	d	$d-d_{REF}$	s
01	-0,00040	0,00040	0,00000	-0,00040	0,00040	0,00000	-0,00050	0,00050	0,00008	-0,00050	0,00050	0,00006
02	0,00930	0,00030	0,00000	0,00930	0,00030	0,00000	0,00930	0,00030	0,00004	0,00930	0,00030	0,00006
03	0,01908	0,00012	0,00004	0,01910	0,00010	0,00000	0,01910	0,00010	0,00000	0,01910	0,00010	0,00004
04	0,02900	0,00020	0,00000	0,02900	0,00020	0,00000	0,02900	0,00020	0,00004	0,02900	0,00020	0,00004
05	0,03896	0,00024	0,00005	0,03894	0,00026	0,00005	0,03890	0,00030	0,00005	0,03890	0,00030	0,00005
06	0,04894	0,00016	0,00005	0,04896	0,00014	0,00005	0,04900	0,00010	0,00000	0,04900	0,00010	0,00004
07	0,05904	0,00006	0,00005	0,05902	0,00008	0,00004	0,05910	0,00000	0,00005	0,05910	0,00000	0,00004
08	0,06900	0,00010	0,00000	0,06902	0,00008	0,00004	0,06910	0,00000	0,00000	0,06910	0,00000	0,00004
09	0,07908	0,00008	0,00004	0,07910	0,00010	0,00000	0,07910	0,00010	0,00005	0,07910	0,00010	0,00004
10	0,08904	0,00004	0,00005	0,08906	0,00006	0,00005	0,08900	0,00000	0,00005	0,08900	0,00000	0,00005
11	0,09912	0,00002	0,00004	0,09910	0,00000	0,00000	0,09910	0,00000	0,00005	0,09910	0,00000	0,00001
12	0,10910	0,00010	0,00000	0,10908	0,00008	0,00004	0,10910	0,00010	0,00004	0,10910	0,00010	0,00005
13	0,11910	0,00010	0,00000	0,11910	0,00010	0,00000	0,11910	0,00010	0,00000	0,11910	0,00010	0,00000
14	0,12916	0,00016	0,00005	0,12920	0,00020	0,00000	0,12910	0,00010	0,00004	0,12910	0,00010	0,00004
15	0,13918	0,00018	0,00004	0,13916	0,00016	0,00005	0,13920	0,00020	0,00005	0,13920	0,00020	0,00000
16	0,14922	0,00022	0,00004	0,14922	0,00022	0,00004	0,14920	0,00020	0,00005	0,14920	0,00020	0,00000
17	0,15924	0,00034	0,00005	0,15922	0,00032	0,00004	0,15920	0,00030	0,00000	0,15920	0,00030	0,00004
18	0,16920	0,00020	0,00000	0,16920	0,00020	0,00000	0,16920	0,00020	0,00004	0,16920	0,00020	0,00000
19	0,17930	0,00030	0,00000	0,17926	0,00026	0,00005	0,17930	0,00030	0,00000	0,17930	0,00030	0,00000
20	0,18930	0,00020	0,00000	0,18930	0,00020	0,00000	0,18930	0,00020	0,00004	0,18930	0,00020	0,00000
21	0,19940	0,00020	0,00000	0,19940	0,00020	0,00000	0,19940	0,00020	0,00000	0,19940	0,00020	0,00000
22	0,20930	0,00010	0,00000	0,20932	0,00012	0,00004	0,20940	0,00020	0,00005	0,20940	0,00020	0,00005
23	0,21930	0,00010	0,00000	0,21930	0,00010	0,00000	0,21930	0,00010	0,00004	0,21930	0,00010	0,00000
24	0,22938	0,00018	0,00004	0,22938	0,00018	0,00004	0,22940	0,00020	0,00004	0,22940	0,00020	0,00004
25	0,23930	0,00010	0,00000	0,23930	0,00010	0,00000	0,23930	0,00010	0,00000	0,23930	0,00010	0,00004
26	0,24936	0,00004	0,00005	0,24936	0,00004	0,00005	0,24940	0,00000	0,00005	0,24940	0,00000	0,00000
27	0,25934	0,00004	0,00005	0,25936	0,00006	0,00005	0,25940	0,00010	0,00005	0,25940	0,00010	0,00004
28	0,26930	0,00000	0,00000	0,26930	0,00000	0,00000	0,26930	0,00000	0,00005	0,26930	0,00000	0,00000
29	0,27934	0,00004	0,00005	0,27936	0,00006	0,00005	0,27940	0,00010	0,00005	0,27940	0,00010	0,00004
30	0,28930	0,00010	0,00000	0,28930	0,00010	0,00000	0,28930	0,00010	0,00005	0,28930	0,00010	0,00004
31	0,29940	0,00010	0,00000	0,29944	0,00006	0,00005	0,29950	0,00000	0,00005	0,29950	0,00000	0,00005
32	0,30930	0,00000	0,00000	0,30930	0,00000	0,00000	0,30930	0,00000	0,00004	0,30930	0,00000	0,00005
33	0,31928	0,00008	0,00004	0,31926	0,00006	0,00005	0,31920	0,00000	0,00005	0,31920	0,00000	0,00005
34	0,32930	0,00000	0,00000	0,32930	0,00000	0,00000	0,32930	0,00000	0,00004	0,32930	0,00000	0,00000
35	0,33922	0,00018	0,00004	0,33922	0,00018	0,00004	0,33930	0,00010	0,00004	0,33930	0,00010	0,00004
36	0,34924	0,00006	0,00005	0,34924	0,00006	0,00005	0,34920	0,00010	0,00005	0,34920	0,00010	0,00004
37	0,35920	0,00000	0,00000	0,35920	0,00000	0,00000	0,35920	0,00000	0,00000	0,35920	0,00000	0,00000
38	0,36910	0,00010	0,00000	0,36910	0,00010	0,00000	0,36910	0,00010	0,00004	0,36910	0,00010	0,00000
39	0,37912	0,00008	0,00004	0,37914	0,00006	0,00005	0,37920	0,00000	0,00005	0,37920	0,00000	0,00006
40	0,38916	0,00004	0,00005	0,38916	0,00004	0,00005	0,38910	0,00010	0,00008	0,38910	0,00010	0,00004
41	0,39930	0,00010	0,00000	0,39930	0,00010	0,00000	0,39930	0,00010	0,00000	0,39930	0,00010	0,00000
42	0,40920	0,00000	0,00000	0,40920	0,00000	0,00000	0,40920	0,00000	0,00004	0,40920	0,00000	0,00005
43	0,41920	0,00010	0,00000	0,41920	0,00010	0,00000	0,41920	0,00010	0,00000	0,41920	0,00010	0,00000
44	0,42920	0,00010	0,00000	0,42920	0,00010	0,00000	0,42920	0,00010	0,00005	0,42920	0,00010	0,00001
45	0,43908	0,00012	0,00004	0,43910	0,00010	0,00000	0,43900	0,00020	0,00000	0,43900	0,00020	0,00005
46	0,44910	0,00010	0,00000	0,44910	0,00010	0,00000	0,44910	0,00010	0,00004	0,44910	0,00010	0,00005
47	0,45910	0,00010	0,00000	0,45910	0,00010	0,00000	0,45910	0,00010	0,00000	0,45910	0,00010	0,00005
48	0,46910	0,00010	0,00000	0,46910	0,00010	0,00000	0,46910	0,00010	0,00000	0,46910	0,00010	0,00005
49	0,47916	0,00004	0,00005	0,47914	0,00006	0,00005	0,47910	0,00010	0,00005	0,47910	0,00010	0,00004
50	0,48910	0,00020	0,00000	0,48906	0,00024	0,00005	0,48900	0,00030	0,00005	0,48900	0,00030	0,00004
51	0,49920	0,00010	0,00000	0,49918	0,00012	0,00004	0,49920	0,00010	0,00000	0,49920	0,00010	0,00001
		σ_E	σ_R		σ_E	σ_R		σ_E	σ_R		σ_E	σ_R
		0,000090	0,000022		0,000089	0,000024		0,000106	0,000037		0,000106	0,000032

Tabela 5-31: Influência do detector de bordas sobre a repetibilidade e exatidão das medidas.

Reconstruindo-se as imagens com o dobro de resolução, utilizando-se para tanto o filtro proposto por NALWA (1987), obtêm-se traços com pouco mais do que 4 pixels de espessura, e, com isso, um domínio mais adequado à aplicação dos detectores de bordas baseados em máscaras 3x3 ou 5x5.

Conforme se pode observar na tabela 5-32, os experimentos realizados com as imagens reconstruídas demonstraram que, para os três detectores testados, houve um certo incremento na exatidão das medidas acompanhado por perda em repetibilidade. Tal fato pode ser explicado pela combinação de dois efeitos: eliminação do viés devido à largura da máscara e amplificação dos ruídos da imagem pelo filtro de reconstrução.

		Sobel			Jensen			Lyvers		
T	d_{REF}	d	$d-d_{REF}$	s	d	$d-d_{REF}$	s	d	$d-d_{REF}$	s
01	0,0000	-0,00034	0,00034	0,00005	-0,00030	0,00030	0,00014	-0,00033	0,00033	0,00008
02	0,0096	0,00930	0,00030	0,00000	0,00934	0,00026	0,00005	0,00938	0,00023	0,00031
03	0,0192	0,01910	0,00010	0,00000	0,01910	0,00010	0,00014	0,01930	0,00010	0,00035
04	0,0292	0,02902	0,00018	0,00004	0,02902	0,00018	0,00018	0,02900	0,00020	0,00016
05	0,0392	0,03896	0,00024	0,00005	0,03884	0,00036	0,00005	0,03905	0,00015	0,00015
06	0,0491	0,04896	0,00014	0,00005	0,04888	0,00022	0,00004	0,04900	0,00010	0,00019
07	0,0591	0,05904	0,00006	0,00005	0,05910	0,00000	0,00019	0,05895	0,00015	0,00011
08	0,0691	0,06900	0,00010	0,00000	0,06894	0,00016	0,00005	0,06893	0,00017	0,00015
09	0,0790	0,07908	0,00008	0,00004	0,07898	0,00002	0,00004	0,07903	0,00002	0,00015
10	0,0890	0,08902	0,00002	0,00004	0,08904	0,00004	0,00005	0,08913	0,00013	0,00008
11	0,0991	0,09912	0,00002	0,00004	0,09904	0,00006	0,00005	0,09903	0,00007	0,00022
12	0,1090	0,10910	0,00010	0,00000	0,10908	0,00008	0,00008	0,10930	0,00030	0,00031
13	0,1190	0,11910	0,00010	0,00000	0,11910	0,00010	0,00010	0,11908	0,00008	0,00023
14	0,1290	0,12918	0,00018	0,00004	0,12914	0,00014	0,00005	0,12920	0,00020	0,00007
15	0,1390	0,13916	0,00016	0,00005	0,13914	0,00014	0,00005	0,13903	0,00002	0,00015
16	0,1490	0,14920	0,00020	0,00000	0,14922	0,00022	0,00008	0,14938	0,00037	0,00020
17	0,1589	0,15920	0,00030	0,00000	0,15924	0,00034	0,00005	0,15908	0,00018	0,00023
18	0,1690	0,16920	0,00020	0,00000	0,16924	0,00024	0,00005	0,16918	0,00018	0,00011
19	0,1790	0,17930	0,00030	0,00000	0,17930	0,00030	0,00007	0,17913	0,00013	0,00027
20	0,1891	0,18930	0,00020	0,00000	0,18934	0,00024	0,00005	0,18940	0,00030	0,00019
21	0,1992	0,19940	0,00020	0,00000	0,19942	0,00022	0,00004	0,19925	0,00005	0,00023
22	0,2092	0,20930	0,00010	0,00000	0,20936	0,00016	0,00009	0,20928	0,00007	0,00015
23	0,2192	0,21930	0,00010	0,00000	0,21936	0,00016	0,00009	0,21913	0,00008	0,00015
24	0,2292	0,22936	0,00016	0,00005	0,22938	0,00018	0,00011	0,22948	0,00027	0,00013
25	0,2392	0,23930	0,00010	0,00000	0,23930	0,00010	0,00014	0,23918	0,00002	0,00023
26	0,2494	0,24936	0,00004	0,00005	0,24934	0,00006	0,00013	0,24928	0,00013	0,00011
27	0,2593	0,25936	0,00006	0,00005	0,25936	0,00006	0,00015	0,25923	0,00007	0,00022
28	0,2693	0,26924	0,00006	0,00005	0,26932	0,00002	0,00011	0,26938	0,00007	0,00015
29	0,2793	0,27934	0,00004	0,00005	0,27938	0,00008	0,00013	0,27925	0,00005	0,00022
30	0,2894	0,28930	0,00010	0,00000	0,28924	0,00016	0,00017	0,28928	0,00012	0,00016
31	0,2995	0,29942	0,00008	0,00004	0,29946	0,00004	0,00011	0,29935	0,00015	0,00022
32	0,3093	0,30930	0,00000	0,00000	0,30924	0,00006	0,00005	0,30935	0,00005	0,00015
33	0,3192	0,31924	0,00004	0,00005	0,31922	0,00002	0,00008	0,31920	0,00000	0,00012
34	0,3293	0,32930	0,00000	0,00000	0,32924	0,00006	0,00005	0,32923	0,00007	0,00015
35	0,3394	0,33922	0,00018	0,00004	0,33924	0,00016	0,00005	0,33923	0,00017	0,00015
36	0,3493	0,34924	0,00006	0,00005	0,34924	0,00006	0,00005	0,34930	0,00000	0,00016
37	0,3592	0,35920	0,00000	0,00000	0,35924	0,00004	0,00005	0,35918	0,00002	0,00015
38	0,3692	0,36910	0,00010	0,00000	0,36918	0,00002	0,00011	0,36913	0,00007	0,00016
39	0,3792	0,37914	0,00006	0,00005	0,37912	0,00008	0,00011	0,37910	0,00010	0,00007
40	0,3892	0,38918	0,00002	0,00004	0,38916	0,00004	0,00015	0,38928	0,00007	0,00031
41	0,3992	0,39932	0,00012	0,00004	0,39940	0,00020	0,00010	0,39920	0,00000	0,00007
42	0,4092	0,40924	0,00004	0,00005	0,40922	0,00002	0,00011	0,40923	0,00002	0,00008
43	0,4191	0,41920	0,00010	0,00000	0,41922	0,00012	0,00011	0,41925	0,00015	0,00011
44	0,4291	0,42920	0,00010	0,00000	0,42922	0,00012	0,00011	0,42925	0,00015	0,00011
45	0,4392	0,43908	0,00012	0,00004	0,43900	0,00020	0,00010	0,43905	0,00015	0,00009
46	0,4492	0,44910	0,00010	0,00000	0,44910	0,00010	0,00012	0,44928	0,00007	0,00031
47	0,4592	0,45910	0,00010	0,00000	0,45904	0,00016	0,00005	0,45915	0,00005	0,00009
48	0,4692	0,46908	0,00012	0,00004	0,46904	0,00016	0,00005	0,46918	0,00002	0,00015
49	0,4792	0,47918	0,00002	0,00004	0,47914	0,00006	0,00005	0,47920	0,00000	0,00007
50	0,4893	0,48910	0,00020	0,00000	0,48914	0,00016	0,00005	0,48925	0,00005	0,00039
51	0,4993	0,49922	0,00008	0,00004	0,49920	0,00010	0,00007	0,49918	0,00012	0,00013
			σ_E	σ_R		σ_E	σ_R		σ_E	σ_R
			0,000083	0,000026		0,000089	0,000090		0,000089	0,000170

Tabela 5-32: Avaliação dos operadores de Sobel, Jensen e Lyvers aplicados a imagens reconstruídas.

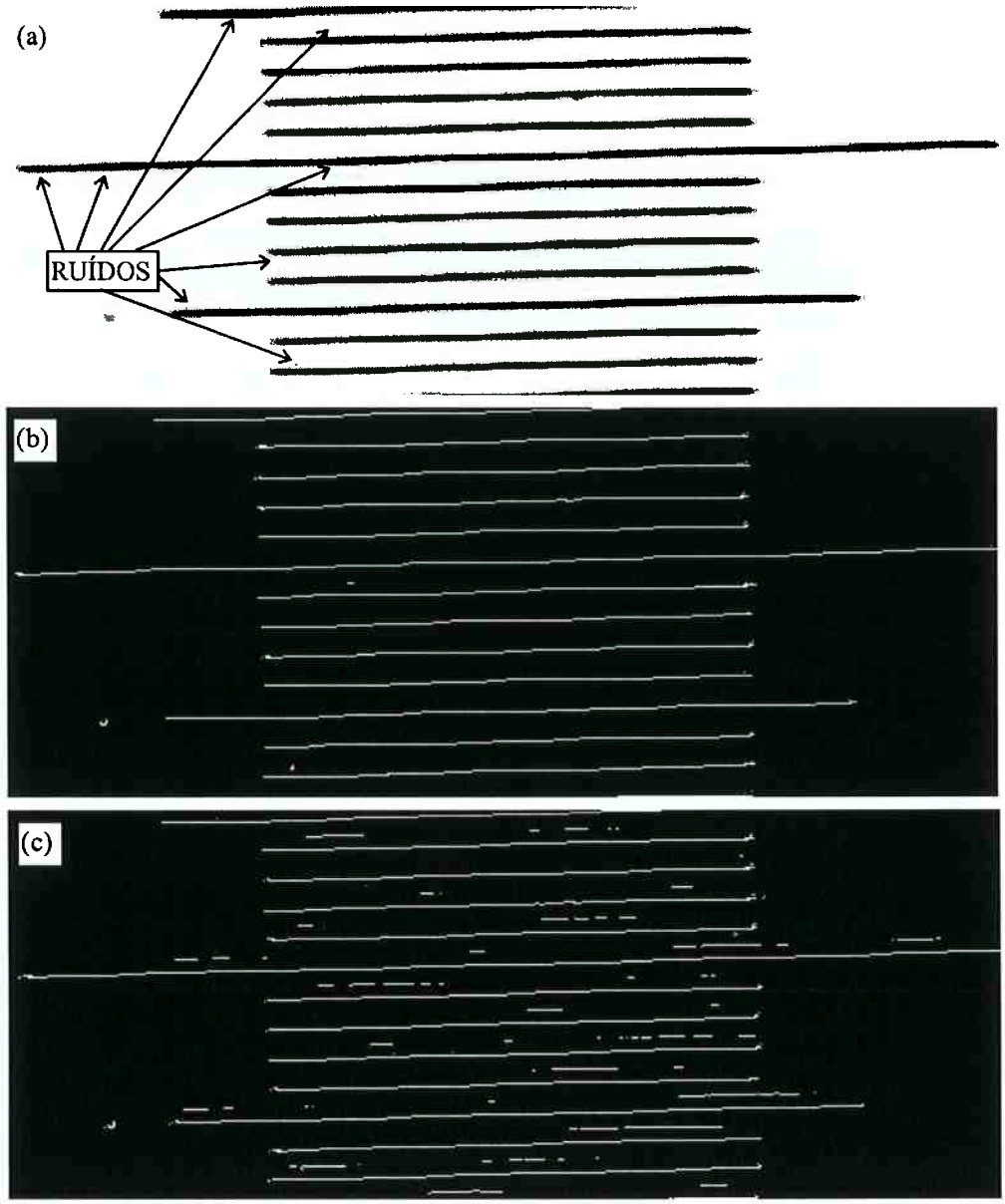

Figura 5.11: (a) Amplificação dos ruídos na imagem reconstruída; (b) Insensibilidade do operador de Sobel à presença de ruídos; (c) Sensibilidade do operador de Lyvers et al à presença de ruídos.

Na figura 5.11 destacam-se na imagem reconstruída as regiões corrompidas pela amplificação de ruído da imagem original. A figura 5.11-c, em particular, salienta as falsas bordas detectadas pelo operador de Lyvers *et al* que, não eliminadas pelo método de limiarização implementado, produzem desvios na localização das retas quando se aplicam as ferramentas de medição em escala ordinal e racional. A despeito de não localizar bordas com precisão de subpixel, o detector de Sobel manteve-se à frente dos

outros dois operadores, pois se mostrou muito mais insensível à presença de ruídos (figura 5.10-b).

T	d_1	d_2	d_3	d_4	d_5	d	s	δ	d_{REF}	e
01	-0,0004	-0,0004	-0,0004	-0,0004	-0,0004	-0,00040	0,00000	0,00000	0,0000	0,00040
02	0,0093	0,0093	0,0093	0,0093	0,0093	0,00930	0,00000	0,00000	0,0096	0,00030
03	0,0191	0,0191	0,0191	0,0191	0,0190	0,01908	0,00004	0,00006	0,0192	0,00012
04	0,0290	0,0290	0,0290	0,0290	0,0290	0,02900	0,00000	0,00000	0,0292	0,00020
05	0,0390	0,0389	0,0389	0,0390	0,0390	0,03896	0,00005	0,00007	0,0392	0,00024
06	0,0489	0,0489	0,0490	0,0489	0,0490	0,04894	0,00005	0,00007	0,0491	0,00016
07	0,0590	0,0590	0,0591	0,0590	0,0591	0,05904	0,00005	0,00007	0,0591	0,00006
08	0,0690	0,0690	0,0690	0,0690	0,0690	0,06900	0,00000	0,00000	0,0691	0,00010
09	0,0791	0,0790	0,0791	0,0791	0,0791	0,07908	0,00004	0,00006	0,0790	0,00008
10	0,0890	0,0890	0,0891	0,0890	0,0891	0,08904	0,00005	0,00007	0,0890	0,00004
11	0,0991	0,0992	0,0991	0,0991	0,0991	0,09912	0,00004	0,00006	0,0991	0,00002
12	0,1091	0,1091	0,1091	0,1091	0,1091	0,10910	0,00000	0,00000	0,1090	0,00010
13	0,1191	0,1191	0,1191	0,1191	0,1191	0,11910	0,00000	0,00000	0,1190	0,00010
14	0,1292	0,1292	0,1291	0,1292	0,1291	0,12916	0,00005	0,00007	0,1290	0,00016
15	0,1391	0,1392	0,1392	0,1392	0,1392	0,13918	0,00004	0,00006	0,1390	0,00018
16	0,1492	0,1492	0,1492	0,1493	0,1492	0,14922	0,00004	0,00006	0,1490	0,00022
17	0,1593	0,1592	0,1592	0,1593	0,1592	0,15924	0,00005	0,00007	0,1589	0,00034
18	0,1692	0,1692	0,1692	0,1692	0,1692	0,16920	0,00000	0,00000	0,1690	0,00020
19	0,1793	0,1793	0,1793	0,1793	0,1793	0,17930	0,00000	0,00000	0,1790	0,00030
20	0,1893	0,1893	0,1893	0,1893	0,1893	0,18930	0,00000	0,00000	0,1891	0,00020
21	0,1994	0,1994	0,1994	0,1994	0,1994	0,19940	0,00000	0,00000	0,1992	0,00020
22	0,2093	0,2093	0,2093	0,2093	0,2093	0,20930	0,00000	0,00000	0,2092	0,00010
23	0,2193	0,2193	0,2193	0,2193	0,2193	0,21930	0,00000	0,00000	0,2192	0,00010
24	0,2294	0,2294	0,2293	0,2294	0,2294	0,22938	0,00004	0,00006	0,2292	0,00018
25	0,2393	0,2393	0,2393	0,2393	0,2393	0,23930	0,00000	0,00000	0,2392	0,00010
26	0,2494	0,2493	0,2494	0,2494	0,2493	0,24936	0,00005	0,00007	0,2494	0,00004
27	0,2593	0,2593	0,2594	0,2594	0,2593	0,25934	0,00005	0,00007	0,2593	0,00004
28	0,2693	0,2693	0,2693	0,2693	0,2693	0,26930	0,00000	0,00000	0,2693	0,00000
29	0,2793	0,2793	0,2794	0,2793	0,2794	0,27934	0,00005	0,00007	0,2793	0,00004
30	0,2893	0,2893	0,2893	0,2893	0,2893	0,28930	0,00000	0,00000	0,2894	0,00010
31	0,2994	0,2994	0,2994	0,2994	0,2994	0,29940	0,00000	0,00000	0,2995	0,00010
32	0,3093	0,3093	0,3093	0,3093	0,3093	0,30930	0,00000	0,00000	0,3093	0,00000
33	0,3193	0,3193	0,3192	0,3193	0,3193	0,31928	0,00004	0,00006	0,3192	0,00008
34	0,3293	0,3293	0,3293	0,3293	0,3293	0,32930	0,00000	0,00000	0,3293	0,00000
35	0,3392	0,3392	0,3393	0,3392	0,3392	0,33922	0,00004	0,00006	0,3394	0,00018
36	0,3493	0,3492	0,3493	0,3492	0,3492	0,34924	0,00005	0,00007	0,3493	0,00006
37	0,3592	0,3592	0,3592	0,3592	0,3592	0,35920	0,00000	0,00000	0,3592	0,00000
38	0,3691	0,3691	0,3691	0,3691	0,3691	0,36910	0,00000	0,00000	0,3692	0,00010
39	0,3791	0,3792	0,3791	0,3791	0,3791	0,37912	0,00004	0,00006	0,3792	0,00008
40	0,3892	0,3891	0,3892	0,3892	0,3891	0,38916	0,00005	0,00007	0,3892	0,00004
41	0,3993	0,3993	0,3993	0,3993	0,3993	0,39930	0,00000	0,00000	0,3992	0,00010
42	0,4092	0,4092	0,4092	0,4092	0,4092	0,40920	0,00000	0,00000	0,4092	0,00000
43	0,4192	0,4192	0,4192	0,4192	0,4192	0,41920	0,00000	0,00000	0,4191	0,00010
44	0,4292	0,4292	0,4292	0,4292	0,4292	0,42920	0,00000	0,00000	0,4291	0,00010
45	0,4391	0,4391	0,4390	0,4391	0,4391	0,43908	0,00004	0,00006	0,4392	0,00012
46	0,4491	0,4491	0,4491	0,4491	0,4491	0,44910	0,00000	0,00000	0,4492	0,00010
47	0,4591	0,4591	0,4591	0,4591	0,4591	0,45910	0,00000	0,00000	0,4592	0,00010
48	0,4691	0,4691	0,4691	0,4691	0,4691	0,46910	0,00000	0,00000	0,4692	0,00010
49	0,4792	0,4792	0,4791	0,4792	0,4791	0,47916	0,00005	0,00007	0,4792	0,00004
50	0,4891	0,4891	0,4891	0,4891	0,4891	0,48910	0,00000	0,00000	0,4893	0,00020
51	0,4992	0,4992	0,4992	0,4992	0,4992	0,49920	0,00000	0,00000	0,4993	0,00010
							σ_R			σ_E
							0,000022			0,000090

Tabela 5-33: Medição do desempenho do IMVC-E_4 em sua condição ótima de operação.

Na tabela 5-33, finalmente, apresentam-se as medidas, incertezas e estimativas de exatidão e repetibilidade do IMVC-E_4, ajustado para seu ponto ideal de operação que, com base nas experiências e considerações apresentadas anteriormente, fica assim definido: escala nominal: *largura do filtro de suavização* = 3; *tipo de borda* = medial;

reconstrução = ausente; *tipo de detector* = Roberts [9]; escala ordinal: *transformada de Hough* ($d\theta$=0,50; s_ρ=6; L=150; E=1); escala racional: *ajuste por mínimos quadrados*. É imediato observar que esse instrumento possui um desempenho notável, registrando incerteza média da ordem de 1/10 da que se obtém através do método convencional.

5.5.5 Medição do Desempenho do IMVC-E_5

O ajuste e avaliação desse instrumento foi realizado a partir de experimentos com as 5 imagens da série 88,5°, adotando-se para as escalas ordinal e racional os mesmos parâmetros ótimos utilizados nos demais instrumentos baseados no método evolutivo.

Com relação à escala nominal, foram testados os dois métodos de medição implementados, a saber: seleção dos *pontos de mínimo* dos traços previamente rotulados da imagem em tons de cinza e aplicação de *detectores especiais de bordas tipo linha*, construídos a partir de perfis médios de variação dos níveis de cinza ao longo da direção normal aos três tipos de traços [10] observados nas imagens de microescalas.

Nas tabelas 5-34 e 5-35 acima são reportados os resultados desses experimentos, observando-se então o impacto positivo sobre a repetibilidade do IMVC-E_5 quando da utilização de ferramenta de medição em escala nominal baseada nos referidos detectores especiais de bordas tipo *linha* em lugar de se realizar a simples seleção dos mínimos locais da imagem em tons de cinza. Também nesse caso, o novo instrumento, apesar de não apresentar repetibilidade tão elevada quanto o IMVC-E_3 e o IMVC-E_4, demonstra um desempenho quase 3 vezes superior ao ensaio convencional.

---

[9] Apesar da pequena vantagem obtida pelo operador de Sobel aplicado a imagens reconstruídas, face ao operador de Roberts aplicado a imagens de baixa resolução, a escolha deste último se deve ao enorme ganho em tempo de processamento obtido quando se aplica o IMVC-E_4 a imagens de baixa resolução.

[10] isto é, a intervalos de 0,10mm (*longos*), 0,05mm (*médios*) e 0,01mm (*curtos*).

T	d_1	d_2	d_3	d_4	d_5	d	s	δ	d_{REF}	e
01	-0,0003	-0,0004	-0,0005	-0,0005	-0,0005	-0,00044	0,00009	0,00011	0,0000	0,00044
02	0,0092	0,0092	0,0094	0,0092	0,0092	0,00924	0,00009	0,00011	0,0096	0,00036
03	0,0191	0,0191	0,0191	0,0190	0,0190	0,01906	0,00005	0,00007	0,0192	0,00014
04	0,0289	0,0291	0,0289	0,0290	0,0290	0,02898	0,00008	0,00010	0,0292	0,00022
05	0,0390	0,0388	0,0389	0,0388	0,0389	0,03888	0,00008	0,00010	0,0392	0,00032
06	0,0489	0,0489	0,0489	0,0489	0,0490	0,04892	0,00004	0,00006	0,0491	0,00018
07	0,0589	0,0590	0,0589	0,0590	0,0591	0,05898	0,00008	0,00010	0,0591	0,00012
08	0,0689	0,0690	0,0689	0,0689	0,0688	0,06890	0,00007	0,00009	0,0691	0,00020
09	0,0790	0,0790	0,0790	0,0791	0,0791	0,07904	0,00005	0,00007	0,0790	0,00004
10	0,0890	0,0890	0,0891	0,0889	0,0888	0,08896	0,00011	0,00014	0,0890	0,00004
11	0,0993	0,0992	0,0990	0,0991	0,0991	0,09914	0,00011	0,00014	0,0991	0,00004
12	0,1091	0,1091	0,1090	0,1090	0,1091	0,10906	0,00005	0,00007	0,1090	0,00006
13	0,1193	0,1191	0,1192	0,1192	0,1190	0,11916	0,00011	0,00014	0,1190	0,00016
14	0,1292	0,1293	0,1291	0,1293	0,1293	0,12924	0,00009	0,00011	0,1290	0,00024
15	0,1392	0,1392	0,1393	0,1391	0,1393	0,13922	0,00008	0,00010	0,1390	0,00022
16	0,1492	0,1492	0,1491	0,1494	0,1493	0,14924	0,00011	0,00014	0,1490	0,00024
17	0,1592	0,1592	0,1592	0,1594	0,1594	0,15928	0,00011	0,00014	0,1589	0,00038
18	0,1691	0,1692	0,1694	0,1692	0,1694	0,16926	0,00013	0,00017	0,1690	0,00026
19	0,1793	0,1792	0,1795	0,1794	0,1792	0,17932	0,00013	0,00016	0,1790	0,00032
20	0,1893	0,1893	0,1892	0,1893	0,1893	0,18928	0,00004	0,00006	0,1891	0,00018
21	0,1994	0,1994	0,1993	0,1993	0,1996	0,19940	0,00012	0,00015	0,1992	0,00020
22	0,2094	0,2094	0,2093	0,2095	0,2093	0,20938	0,00008	0,00010	0,2092	0,00018
23	0,2193	0,2194	0,2193	0,2193	0,2193	0,21932	0,00004	0,00006	0,2192	0,00012
24	0,2293	0,2294	0,2294	0,2294	0,2294	0,22938	0,00004	0,00006	0,2292	0,00018
25	0,2393	0,2393	0,2393	0,2393	0,2393	0,23930	0,00000	0,00000	0,2392	0,00010
26	0,2494	0,2492	0,2494	0,2496	0,2494	0,24940	0,00014	0,00018	0,2494	0,00000
27	0,2594	0,2592	0,2594	0,2596	0,2594	0,25940	0,00014	0,00018	0,2593	0,00010
28	0,2692	0,2692	0,2693	0,2694	0,2693	0,26928	0,00008	0,00010	0,2693	0,00002
29	0,2792	0,2794	0,2797	0,2794	0,2795	0,27944	0,00018	0,00023	0,2793	0,00014
30	0,2895	0,2895	0,2894	0,2893	0,2892	0,28938	0,00013	0,00016	0,2894	0,00002
31	0,2995	0,2993	0,2993	0,2995	0,2995	0,29942	0,00011	0,00014	0,2995	0,00008
32	0,3095	0,3093	0,3094	0,3092	0,3094	0,30936	0,00011	0,00014	0,3093	0,00006
33	0,3193	0,3195	0,3193	0,3192	0,3192	0,31930	0,00012	0,00015	0,3192	0,00010
34	0,3293	0,3293	0,3292	0,3292	0,3292	0,32924	0,00005	0,00007	0,3293	0,00006
35	0,3393	0,3391	0,3392	0,3394	0,3392	0,33924	0,00011	0,00014	0,3394	0,00016
36	0,3492	0,3493	0,3493	0,3493	0,3492	0,34926	0,00005	0,00007	0,3493	0,00004
37	0,3591	0,3591	0,3594	0,3591	0,3592	0,35918	0,00013	0,00016	0,3592	0,00002
38	0,3691	0,3692	0,3692	0,3691	0,3692	0,36916	0,00005	0,00007	0,3692	0,00004
39	0,3790	0,3791	0,3792	0,3791	0,3792	0,37912	0,00008	0,00010	0,3792	0,00008
40	0,3892	0,3890	0,3893	0,3893	0,3892	0,38920	0,00012	0,00015	0,3892	0,00000
41	0,3994	0,3994	0,3993	0,3994	0,3992	0,39934	0,00009	0,00011	0,3992	0,00014
42	0,4091	0,4091	0,4094	0,4091	0,4093	0,40920	0,00014	0,00018	0,4092	0,00000
43	0,4191	0,4190	0,4193	0,4192	0,4191	0,41914	0,00011	0,00014	0,4191	0,00004
44	0,4289	0,4292	0,4292	0,4291	0,4293	0,42914	0,00015	0,00019	0,4291	0,00004
45	0,4393	0,4392	0,4391	0,4390	0,4390	0,43912	0,00013	0,00016	0,4392	0,00008
46	0,4491	0,4492	0,4490	0,4490	0,4491	0,44908	0,00008	0,00010	0,4492	0,00012
47	0,4591	0,4592	0,4591	0,4592	0,4591	0,45914	0,00005	0,00007	0,4592	0,00006
48	0,4691	0,4691	0,4690	0,4689	0,4692	0,46906	0,00011	0,00014	0,4692	0,00014
49	0,4792	0,4792	0,4792	0,4791	0,4790	0,47914	0,00009	0,00011	0,4792	0,00006
50	0,4890	0,4891	0,4889	0,4891	0,4890	0,48902	0,00008	0,00010	0,4893	0,00028
51	0,4992	0,4990	0,4992	0,4991	0,4991	0,49912	0,00008	0,00010	0,4993	0,00018
							σ_R			σ_E
							0,000095			0,000106

Tabela 5.34: Desempenho do IMVC-E_5 usando os pontos de mínimo da imagem em tons de cinza.

T	d_1	d_2	d_3	d_4	d_5	d	s	δ	d_{REF}	e
01	-0,0003	-0,0003	-0,0004	-0,0004	-0,0004	-0,00036	0,00005	0,00007	0,0000	0,00036
02	0,0092	0,0092	0,0094	0,0093	0,0093	0,00928	0,00008	0,00010	0,0096	0,00032
03	0,0189	0,0191	0,0191	0,0190	0,0190	0,01902	0,00008	0,00010	0,0192	0,00018
04	0,0291	0,0291	0,0289	0,0290	0,0290	0,02902	0,00008	0,00010	0,0292	0,00018
05	0,0389	0,0390	0,0389	0,0388	0,0388	0,03888	0,00008	0,00010	0,0392	0,00032
06	0,0489	0,0489	0,0488	0,0488	0,0488	0,04884	0,00005	0,00007	0,0491	0,00026
07	0,0591	0,0590	0,0589	0,0589	0,0589	0,05896	0,00009	0,00011	0,0591	0,00014
08	0,0691	0,0690	0,0689	0,0689	0,0689	0,06896	0,00009	0,00011	0,0691	0,00014
09	0,0790	0,0790	0,0789	0,0790	0,0790	0,07898	0,00004	0,00006	0,0790	0,00002
10	0,0890	0,0890	0,0890	0,0891	0,0890	0,08902	0,00004	0,00006	0,0890	0,00002
11	0,0991	0,0991	0,0990	0,0990	0,0990	0,09904	0,00005	0,00007	0,0991	0,00006
12	0,1091	0,1091	0,1090	0,1091	0,1091	0,10908	0,00004	0,00006	0,1090	0,00008
13	0,1192	0,1191	0,1191	0,1192	0,1192	0,11916	0,00005	0,00007	0,1190	0,00016
14	0,1293	0,1292	0,1292	0,1293	0,1293	0,12926	0,00005	0,00007	0,1290	0,00026
15	0,1392	0,1392	0,1393	0,1392	0,1393	0,13924	0,00005	0,00007	0,1390	0,00024
16	0,1493	0,1492	0,1493	0,1494	0,1493	0,14930	0,00007	0,00009	0,1490	0,00030
17	0,1592	0,1593	0,1593	0,1594	0,1594	0,15932	0,00008	0,00010	0,1589	0,00042
18	0,1692	0,1692	0,1693	0,1692	0,1693	0,16924	0,00005	0,00007	0,1690	0,00024
19	0,1793	0,1793	0,1794	0,1794	0,1794	0,17936	0,00005	0,00007	0,1790	0,00036
20	0,1893	0,1893	0,1894	0,1893	0,1893	0,18932	0,00004	0,00006	0,1891	0,00022
21	0,1994	0,1994	0,1993	0,1995	0,1995	0,19942	0,00008	0,00010	0,1992	0,00022
22	0,2093	0,2094	0,2093	0,2093	0,2093	0,20932	0,00004	0,00006	0,2092	0,00012
23	0,2193	0,2192	0,2193	0,2193	0,2193	0,21928	0,00004	0,00006	0,2192	0,00008
24	0,2294	0,2293	0,2294	0,2294	0,2294	0,22938	0,00004	0,00006	0,2292	0,00018
25	0,2392	0,2392	0,2393	0,2393	0,2393	0,23926	0,00005	0,00007	0,2392	0,00006
26	0,2493	0,2493	0,2494	0,2494	0,2494	0,24936	0,00005	0,00007	0,2494	0,00004
27	0,2592	0,2592	0,2594	0,2595	0,2594	0,25934	0,00013	0,00017	0,2593	0,00004
28	0,2694	0,2694	0,2693	0,2694	0,2693	0,26936	0,00005	0,00007	0,2693	0,00006
29	0,2792	0,2793	0,2795	0,2793	0,2794	0,27934	0,00011	0,00014	0,2793	0,00004
30	0,2895	0,2893	0,2894	0,2892	0,2893	0,28934	0,00011	0,00014	0,2894	0,00006
31	0,2994	0,2993	0,2995	0,2995	0,2994	0,29942	0,00008	0,00010	0,2995	0,00008
32	0,3093	0,3093	0,3092	0,3092	0,3093	0,30926	0,00005	0,00007	0,3093	0,00004
33	0,3193	0,3193	0,3192	0,3192	0,3192	0,31924	0,00005	0,00007	0,3192	0,00004
34	0,3293	0,3293	0,3292	0,3292	0,3292	0,32924	0,00005	0,00007	0,3293	0,00006
35	0,3392	0,3392	0,3392	0,3394	0,3393	0,33926	0,00009	0,00011	0,3394	0,00014
36	0,3493	0,3493	0,3492	0,3493	0,3494	0,34930	0,00007	0,00009	0,3493	0,00000
37	0,3593	0,3591	0,3592	0,3592	0,3592	0,35920	0,00007	0,00009	0,3592	0,00000
38	0,3691	0,3690	0,3692	0,3691	0,3692	0,36912	0,00008	0,00010	0,3692	0,00008
39	0,3790	0,3791	0,3792	0,3791	0,3792	0,37912	0,00008	0,00010	0,3792	0,00008
40	0,3891	0,3890	0,3893	0,3893	0,3892	0,38918	0,00013	0,00016	0,3892	0,00002
41	0,3993	0,3993	0,3994	0,3994	0,3994	0,39936	0,00005	0,00007	0,3992	0,00016
42	0,4091	0,4092	0,4093	0,4093	0,4093	0,40924	0,00009	0,00011	0,4092	0,00004
43	0,4191	0,4191	0,4193	0,4192	0,4193	0,41920	0,00010	0,00012	0,4191	0,00010
44	0,4291	0,4293	0,4293	0,4291	0,4291	0,42918	0,00011	0,00014	0,4291	0,00008
45	0,4391	0,4391	0,4390	0,4390	0,4390	0,43904	0,00005	0,00007	0,4392	0,00016
46	0,4491	0,4492	0,4490	0,4490	0,4491	0,44908	0,00008	0,00010	0,4492	0,00012
47	0,4591	0,4592	0,4591	0,4590	0,4590	0,45908	0,00008	0,00010	0,4592	0,00012
48	0,4691	0,4691	0,4690	0,4691	0,4690	0,46906	0,00005	0,00007	0,4692	0,00014
49	0,4792	0,4792	0,4791	0,4791	0,4791	0,47914	0,00005	0,00007	0,4792	0,00006
50	0,4891	0,4890	0,4891	0,4891	0,4891	0,48908	0,00004	0,00006	0,4893	0,00022
51	0,4993	0,4992	0,4992	0,4992	0,4991	0,49920	0,00007	0,00009	0,4993	0,00010
							σ_R			σ_E
							0,000071			0,000105

Tabela 5-35: Desempenho do IMVC-E_5 usando bordas tipo linha detectadas por um operador especial.

5.6 O Modelo de Medição de Microescalas

Conforme já mencionado, dos cinco instrumentos implementados para medição de microescalas, o IMVC-E_1 é o único que se presta tão somente à medição qualitativa. Todavia, sua aplicação em uma bancada metrológica pode ser bastante proveitosa,

especialmente quando se recebem lotes de microescalas do mesmo fabricante. Em tal circunstância, a qualidade do lote pode ser rapidamente avaliada [11]: basta estimar a variação δf_0 da freqüência espacial fundamental das imagens capturadas e verificar se ela se encontra no intervalo de precisão do IMVC-E_1 (±0,00004 μm) para se concluir que o lote apresenta, ou não, qualidade Q^L compatível com a da microescala-referência.

Quanto aos demais instrumentos, o confronto de suas características de exatidão (σ_E) e repetibilidade (σ_R), determinadas através dos experimentos reportados nos ítens 5.5.2 a 5.5.5, revela que o IMVC-E_4 é o que tem melhor figura de mérito relativamente àqueles parâmetros, pois, individualmente, é o que possui maior repetibilidade e, confrontado com o instrumento que registrou o menor valor σ_E (IMVC-E_2), possui cerca de 85% da sua exatidão, mas repetibilidade 9 vezes superior.

5.7 Medições Rotineiras de Microescalas

Existem duas formas de se medir uma nova microescala através do IMVC-E_4: 1ª) utilizando uma microescala-referência (M^R), pré-calibrada pelo processo tradicional; 2ª) utilizando um par de imagens da nova microescala (M^T), deslocada mecanicamente de uma distância D conhecida.

No primeiro caso, é necessário que as imagens de M^R e M^T sejam capturadas em condições bastante similares, o que requer um certo controle do posicionamento angular α dos traços e da distância focal f. Como as microescalas são impressas em pastilhas inseridas em uma base retangular de forma que os traços fiquem aproximadamente paralelos ao lado maior [12], um simples dispositivo mecânico de fixação é suficiente para reproduzir a orientação α dentro de uma tolerância aceitável. Quanto ao controle da

[11] O tempo do ensaio é praticamente igual ao tempo de captura das imagens, pois a freqüência espacial f_0 de cada imagem é calculada em menos de 1 seg, utilizando-se um computador PC-Pentium de 70 MHz.

distância focal, basta utilizar blocos-padrão para compensar variações de espessura superiores a 0,06 mm.

Adotadas as providências acima, a medição é realizada simplesmente aplicando-se às imagens de M^T o mesmo fator de escala K (em *mm/nº de pixels*) determinado pela expressão 3-48, onde o vetor ρ (em *nº de pixels*) é extraído das imagens de M^R e o vetor **r** (em *mm*) corresponde às medidas-referência.

Para se estimar a incerteza desse novo processo de medição, é necessária uma investigação prévia da possível presença de distorções na imagem devidas a deficiências do sistema óptico da câmera. Para tanto, aplicou-se o IMVC-E_4 a três grupos de cinco imagens da microescala situadas nas regiões central, intermediária e extrema do plano-imagem, conforme se apresenta na figura 5.12, calculando-se, para cada grupo, o valor médio $\overline{E}$ dos espaçamentos dos 10 primeiros espaçamentos E_i entre retas sucessivas (ver tabela 5-36).

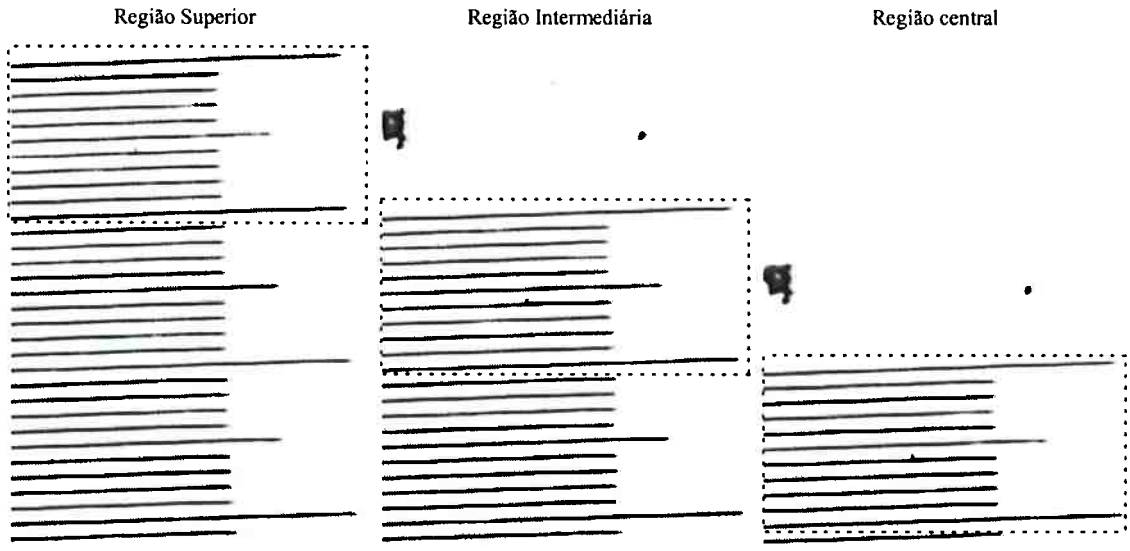

Figura 5.12: Retas posicionadas nas regiões superior, intermediária e central.

[12] De acordo com o Laboratório de Metrologia do IPT, a variação angular é inferior a 1'.

Os resultados da análise de variança apresentados na tabela 5-37 revelam que não se observa, a nível de 95% de significância, correlação da distância média entre retas sucessivas com a particular região do plano-imagem onde elas se situam, ou, o que é equivalente, não há evidências de distorção espacial do plano-imagem.

Retas	Superior E	Intermediária E	Central E
1-2	0,00958	0,00966	0,00956
2-3	0,00984	0,00964	0,00984
3-4	0,00988	0,01002	0,00984
4-5	0,00988	0,00986	0,00998
5-6	0,01004	0,01006	0,00998
6-7	0,01004	0,01002	0,00994
7-8	0,00998	0,00982	0,01000
8-9	0,01006	0,01008	0,00996
9-10	0,00990	0,00998	0,01004
10-11	0,01012	0,01010	0,01010

Tabela 5-36: Variação do espaçamento entre retas em diferentes regiões da imagem.

ANOVA: fator único (região do plano-imagem)

HIPÓTESE H0: $\mu_{Sup} = \mu_{Int} = \mu_{Cen}$

Grupo	*Contagem*	*Soma*	*Média*	*Variância*
Superior	10	0,09932	0,009932	2,40178E-08
Intermediária	10	0,09924	0,009924	2,896E-08
Central	10	0,09924	0,009924	2,27378E-08

Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Entre grupos	4,26667E-10	2	2,13333E-10	0,008452688	0,991585558	3,354131195
Dentro dos grupos	6,8144E-07	27	2,52385E-08			
Total	6,81867E-07	29				

CONCLUSÃO: $F < F_{crítico} \Rightarrow \mu_{Sup} = \mu_{Int} = \mu_{Cen}$

Tabela 5-37: Teste para verificar existência ou não de distorção na imagem.

Uma vez descartada a possibilidade de distorção, pode-se estimar a incerteza do método proposto utilizando-se apenas as estatísticas de repetibilidade das posições das retas medidas em *nº de pixel,* relativamente ao centro de gravidade da sua distribuição.

Considerando-se que, para a amplificação utilizada, resulta um fator de escala aproximadamente igual a 1 μm/pixel, os dados da tabela 5-38 indicam que a incerteza do novo método é da ordem de ±0,04μm, portanto cerca de 20 vezes inferior à do ensaio convencional (±0,2μm). Por esse motivo, as medidas que nos experimentos anteriores

foram adotadas como referência, perdem necessariamente esse *status*, sendo doravante reportadas apenas para fins documentais.

T	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	s	δ
			(medidas em nº de pixel)				
1	-231,12	-231,13	-231,13	-231,12	-231,13	0,004	0,005
2	-222,26	-222,23	-222,26	-222,24	-222,27	0,017	0,021
3	-213,06	-213,09	-213,07	-213,08	-213,06	0,013	0,016
4	-203,95	-203,97	-203,95	-203,95	-203,95	0,007	0,009
5	-194,69	-194,67	-194,68	-194,68	-194,69	0,011	0,014
6	-185,42	-185,37	-185,41	-185,41	-185,42	0,019	0,024
7	-176,16	-176,15	-176,15	-176,15	-176,16	0,004	0,005
8	-166,83	-166,86	-166,80	-166,83	-166,81	0,020	0,025
9	-157,57	-157,60	-157,59	-157,57	-157,58	0,015	0,019
10	-148,35	-148,32	-148,35	-148,34	-148,36	0,017	0,021
11	-138,94	-138,94	-138,93	-138,93	-138,93	0,005	0,006
12	-129,77	-129,77	-129,78	-129,77	-129,78	0,005	0,006
13	-120,48	-120,44	-120,47	-120,47	-120,48	0,018	0,022
14	-111,11	-111,14	-111,12	-111,12	-111,11	0,011	0,014
15	-101,95	-101,95	-101,96	-101,95	-101,95	0,004	0,005
16	-92,58	-92,59	-92,60	-92,59	-92,59	0,008	0,010
17	-83,32	-83,35	-83,32	-83,33	-83,31	0,014	0,017
18	-74,14	-74,12	-74,14	-74,13	-74,15	0,011	0,014
19	-64,72	-64,72	-64,71	-64,71	-64,72	0,006	0,007
20	-55,51	-55,54	-55,52	-55,52	-55,52	0,010	0,012
21	-46,12	-46,10	-46,12	-46,11	-46,12	0,007	0,009
22	-36,87	-36,92	-36,91	-36,92	-36,89	0,019	0,024
23	-27,71	-27,73	-27,71	-27,70	-27,70	0,009	0,011
24	-18,39	-18,35	-18,37	-18,38	-18,39	0,017	0,021
25	-9,16	-9,20	-9,17	-9,17	-9,17	0,017	0,021
26	0,16	0,16	0,17	0,16	0,16	0,007	0,009
27	9,41	9,44	9,42	9,42	9,42	0,011	0,014
28	18,63	18,58	18,60	18,62	18,63	0,021	0,026
29	27,90	27,91	27,89	27,89	27,89	0,009	0,011
30	37,18	37,16	37,18	37,18	37,18	0,009	0,011
31	46,50	46,49	46,51	46,50	46,50	0,005	0,006
32	55,62	55,66	55,64	55,63	55,62	0,017	0,021
33	64,91	64,85	64,92	64,91	64,90	0,028	0,035
34	74,12	74,11	74,10	74,11	74,11	0,005	0,006
35	83,41	83,41	83,42	83,41	83,41	0,006	0,007
36	92,60	92,66	92,61	92,61	92,60	0,025	0,031
37	101,76	101,82	101,78	101,78	101,76	0,023	0,029
38	111,07	111,03	111,06	111,05	111,07	0,015	0,019
39	120,32	120,30	120,31	120,31	120,31	0,010	0,012
40	129,61	129,63	129,62	129,62	129,61	0,010	0,012
41	139,02	139,01	139,00	139,01	139,01	0,006	0,007
42	148,12	148,10	148,12	148,13	148,13	0,013	0,016
43	157,38	157,41	157,38	157,38	157,39	0,013	0,016
44	166,70	166,64	166,70	166,68	166,70	0,023	0,029
45	175,81	175,85	175,82	175,82	175,83	0,017	0,021
46	185,13	185,14	185,13	185,13	185,14	0,007	0,009
47	194,44	194,42	194,40	194,42	194,44	0,016	0,020
48	203,64	203,71	203,67	203,66	203,66	0,027	0,034
49	213,00	213,00	213,01	213,00	213,01	0,005	0,006
50	222,15	222,16	222,15	222,17	222,17	0,011	0,014
51	231,61	231,60	231,61	231,60	231,61	0,005	0,006
						σ_R	δ_{max}
						0,012	0,035

Tabela 5-38: Estimativa da incerteza do processo de medição utilizando imagem-referência.

Para o segundo processo de medição mencionado, o fator de escala K é também obtido a partir da expressão 3-48, porém, nesse caso, conforme indicado na figura 5.13, ρ é dado pelas distâncias entre pares correspondentes de retas nas imagens deslocadas e

r é o vetor uniforme $\frac{D}{\cos\alpha}[1 \quad \cdots \quad 1]$. Além disso, considerando-se que a resolução do micrômetro do microscópio é de 1,0μm, é necessário incluir no arranjo experimental um sistema de medição a laser capaz de medir os deslocamentos mecânicos com a precisão mínima de 0,10μm.

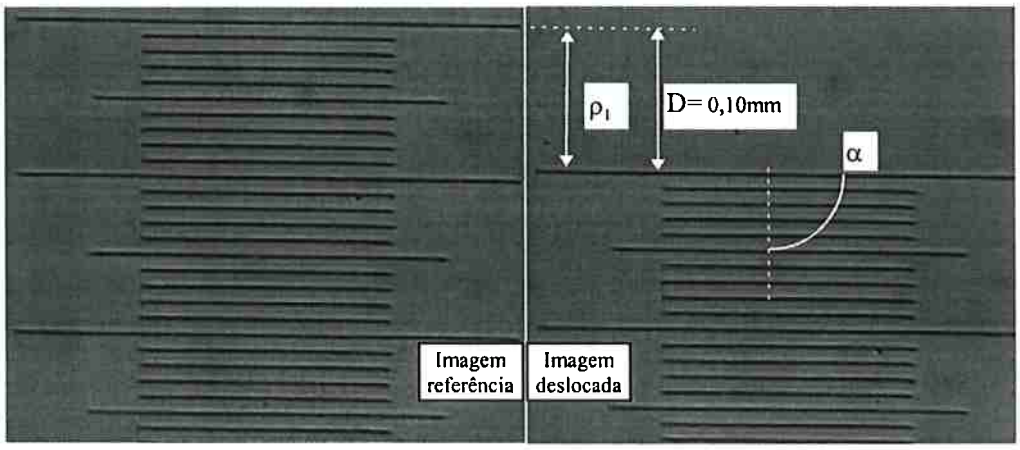

Figura 5.13: Obtenção do fator de escala a partir de deslocamento mecânico.

Para investigar a influência do deslocamento D sobre a qualidade das medidas, utilizaram-se 4 grupos de 5 pares de imagens, deslocadas de distâncias 10, 20, 30 e 40 μm relativamente às imagens-referência (D=0). Para cada um desses grupos, calculou-se o fator de escala médio K, que em seguida foi utilizado para determinar as posições d_{ij} dos traços em cada imagem i, adotando-se como referência o centro de gravidade da distribuição desses traços.

Nas tabelas 5-39 a 5-42 apresentam-se as estimativas de repetibilidade e incerteza das medidas, para os diferentes deslocamentos D selecionados. Todavia, para que se possa adquirir uma melhor compreensão da interação das variáveis que afetam a repetibilidade do instrumento, é importante que se elabore um modelo do processo de propagação de erros de medição.

T	d_{REF}	d_1	d_2	d_3	d_4	d_5	d	s	δ
01	-0,2492	-0,2491	-0,2491	-0,2491	-0,2491	-0,2491	-0,24910	0,00000	0,00000
02	-0,2396	-0,2395	-0,2395	-0,2395	-0,2395	-0,2395	-0,23950	0,00000	0,00000
03	-0,2300	-0,2297	-0,2297	-0,2297	-0,2298	-0,2297	-0,22972	0,00004	0,00006
04	-0,2200	-0,2198	-0,2198	-0,2198	-0,2198	-0,2198	-0,21980	0,00000	0,00000
05	-0,2100	-0,2099	-0,2100	-0,2099	-0,2099	-0,2099	-0,20992	0,00004	0,00006
06	-0,2001	-0,1999	-0,1999	-0,1999	-0,1999	-0,1999	-0,19990	0,00000	0,00000
07	-0,1901	-0,1898	-0,1898	-0,1899	-0,1898	-0,1899	-0,18984	0,00005	0,00007
08	-0,1801	-0,1799	-0,1799	-0,1799	-0,1799	-0,1799	-0,17990	0,00000	0,00000
09	-0,1702	-0,1698	-0,1698	-0,1698	-0,1698	-0,1698	-0,16980	0,00000	0,00000
10	-0,1602	-0,1599	-0,1599	-0,1599	-0,1599	-0,1599	-0,15990	0,00000	0,00000
11	-0,1501	-0,1498	-0,1498	-0,1498	-0,1498	-0,1498	-0,14980	0,00000	0,00000
12	-0,1402	-0,1398	-0,1398	-0,1398	-0,1398	-0,1398	-0,13980	0,00000	0,00000
13	-0,1302	-0,1299	-0,1299	-0,1299	-0,1299	-0,1299	-0,12990	0,00000	0,00000
14	-0,1202	-0,1198	-0,1198	-0,1198	-0,1198	-0,1198	-0,11980	0,00000	0,00000
15	-0,1102	-0,1099	-0,1099	-0,1099	-0,1098	-0,1099	-0,10988	0,00004	0,00006
16	-0,1002	-0,0998	-0,0998	-0,0998	-0,0998	-0,0998	-0,09980	0,00000	0,00000
17	-0,0903	-0,0898	-0,0898	-0,0898	-0,0898	-0,0898	-0,08980	0,00000	0,00000
18	-0,0802	-0,0799	-0,0799	-0,0799	-0,0799	-0,0799	-0,07990	0,00000	0,00000
19	-0,0702	-0,0698	-0,0699	-0,0698	-0,0699	-0,0698	-0,06984	0,00005	0,00007
20	-0,0601	-0,0598	-0,0598	-0,0598	-0,0598	-0,0598	-0,05980	0,00000	0,00000
21	-0,0500	-0,0497	-0,0497	-0,0497	-0,0497	-0,0497	-0,04970	0,00000	0,00000
22	-0,0400	-0,0398	-0,0398	-0,0398	-0,0399	-0,0398	-0,03982	0,00004	0,00006
23	-0,0300	-0,0298	-0,0298	-0,0299	-0,0298	-0,0298	-0,02982	0,00004	0,00006
24	-0,0200	-0,0199	-0,0199	-0,0199	-0,0198	-0,0199	-0,01988	0,00004	0,00006
25	-0,0100	-0,0098	-0,0099	-0,0099	-0,0099	-0,0099	-0,00988	0,00004	0,00006
26	0,0002	0,0002	0,0002	0,0002	0,0001	0,0002	0,00018	0,00004	0,00006
27	0,0101	0,0101	0,0101	0,0101	0,0101	0,0101	0,01010	0,00000	0,00000
28	0,0201	0,0201	0,0201	0,0201	0,0201	0,0201	0,02010	0,00000	0,00000
29	0,0301	0,0301	0,0301	0,0301	0,0301	0,0301	0,03010	0,00000	0,00000
30	0,0402	0,0400	0,0400	0,0401	0,0400	0,0400	0,04002	0,00004	0,00006
31	0,0503	0,0501	0,0501	0,0501	0,0501	0,0501	0,05010	0,00000	0,00000
32	0,0601	0,0600	0,0599	0,0599	0,0599	0,0599	0,05992	0,00004	0,00006
33	0,0700	0,0700	0,0699	0,0700	0,0699	0,0700	0,06996	0,00005	0,00007
34	0,0801	0,0799	0,0800	0,0799	0,0800	0,0799	0,07994	0,00005	0,00007
35	0,0902	0,0898	0,0898	0,0899	0,0898	0,0899	0,08984	0,00005	0,00007
36	0,1001	0,0998	0,0998	0,0998	0,0998	0,0998	0,09980	0,00000	0,00000
37	0,1100	0,1097	0,1097	0,1097	0,1098	0,1097	0,10972	0,00004	0,00006
38	0,1200	0,1197	0,1197	0,1198	0,1197	0,1198	0,11974	0,00005	0,00007
39	0,1300	0,1297	0,1297	0,1297	0,1297	0,1297	0,12970	0,00000	0,00000
40	0,1400	0,1397	0,1397	0,1397	0,1397	0,1397	0,13970	0,00000	0,00000
41	0,1500	0,1498	0,1498	0,1498	0,1498	0,1498	0,14980	0,00000	0,00000
42	0,1600	0,1597	0,1597	0,1597	0,1597	0,1597	0,15970	0,00000	0,00000
43	0,1699	0,1696	0,1696	0,1696	0,1696	0,1696	0,16960	0,00000	0,00000
44	0,1799	0,1797	0,1797	0,1797	0,1797	0,1797	0,17970	0,00000	0,00000
45	0,1900	0,1895	0,1895	0,1895	0,1895	0,1895	0,18950	0,00000	0,00000
46	0,2000	0,1996	0,1996	0,1996	0,1996	0,1996	0,19960	0,00000	0,00000
47	0,2100	0,2096	0,2096	0,2096	0,2096	0,2096	0,20960	0,00000	0,00000
48	0,2200	0,2195	0,2195	0,2195	0,2195	0,2195	0,21950	0,00000	0,00000
49	0,2300	0,2296	0,2296	0,2296	0,2296	0,2296	0,22960	0,00000	0,00000
50	0,2401	0,2395	0,2395	0,2395	0,2395	0,2395	0,23950	0,00000	0,00000
51	0,2501	0,2497	0,2496	0,2496	0,2496	0,2496	0,24962	0,00004	0,00006
								σ_R	δ_{max}
								0,000017	0,00007

Tabela 5-39: Medidas obtidas com $K_{0,100}$ calculado para D = 0,100mm.

T	d_{REF}	d_1	d_2	d_3	d_4	d_5	d	s	δ
01	-0,2492	-0,2494	-0,2494	-0,2494	-0,2494	-0,2494	-0,24940	0,00000	0,00000
02	-0,2396	-0,2398	-0,2398	-0,2397	-0,2398	-0,2397	-0,23976	0,00005	0,00007
03	-0,2300	-0,2299	-0,2300	-0,2300	-0,2299	-0,2300	-0,22996	0,00005	0,00007
04	-0,2200	-0,2200	-0,2200	-0,2200	-0,2200	-0,2200	-0,22000	0,00000	0,00000
05	-0,2100	-0,2102	-0,2102	-0,2102	-0,2101	-0,2101	-0,21016	0,00005	0,00007
06	-0,2001	-0,2001	-0,2001	-0,2001	-0,2001	-0,2001	-0,20010	0,00000	0,00000
07	-0,1901	-0,1900	-0,1900	-0,1900	-0,1900	-0,1900	-0,19000	0,00000	0,00000
08	-0,1801	-0,1801	-0,1801	-0,1801	-0,1801	-0,1801	-0,18010	0,00000	0,00000
09	-0,1702	-0,1700	-0,1699	-0,1699	-0,1700	-0,1699	-0,16994	0,00005	0,00007
10	-0,1602	-0,1601	-0,1601	-0,1601	-0,1601	-0,1601	-0,16010	0,00000	0,00000
11	-0,1501	-0,1499	-0,1499	-0,1499	-0,1499	-0,1499	-0,14990	0,00000	0,00000
12	-0,1402	-0,1400	-0,1400	-0,1399	-0,1399	-0,1399	-0,13994	0,00005	0,00007
13	-0,1302	-0,1301	-0,1300	-0,1301	-0,1300	-0,1301	-0,13006	0,00005	0,00007
14	-0,1202	-0,1199	-0,1199	-0,1199	-0,1199	-0,1199	-0,11990	0,00000	0,00000
15	-0,1102	-0,1100	-0,1100	-0,1099	-0,1100	-0,1099	-0,10996	0,00005	0,00007
16	-0,1002	-0,0999	-0,0999	-0,0999	-0,0999	-0,0999	-0,09990	0,00000	0,00000
17	-0,0903	-0,0899	-0,0899	-0,0899	-0,0899	-0,0899	-0,08990	0,00000	0,00000
18	-0,0802	-0,0800	-0,0800	-0,0799	-0,0800	-0,0799	-0,07996	0,00005	0,00007
19	-0,0702	-0,0699	-0,0699	-0,0699	-0,0699	-0,0699	-0,06990	0,00000	0,00000
20	-0,0601	-0,0598	-0,0598	-0,0598	-0,0598	-0,0598	-0,05980	0,00000	0,00000
21	-0,0500	-0,0498	-0,0498	-0,0498	-0,0498	-0,0498	-0,04980	0,00000	0,00000
22	-0,0400	-0,0398	-0,0399	-0,0399	-0,0398	-0,0399	-0,03986	0,00005	0,00007
23	-0,0300	-0,0299	-0,0298	-0,0298	-0,0298	-0,0298	-0,02982	0,00004	0,00006
24	-0,0200	-0,0199	-0,0199	-0,0198	-0,0199	-0,0199	-0,01988	0,00004	0,00006
25	-0,0100	-0,0099	-0,0099	-0,0099	-0,0099	-0,0099	-0,00990	0,00000	0,00000
26	0,0002	0,0002	0,0001	0,0001	0,0002	0,0001	0,00014	0,00005	0,00007
27	0,0101	0,0101	0,0101	0,0101	0,0101	0,0101	0,01010	0,00000	0,00000
28	0,0201	0,0201	0,0201	0,0201	0,0201	0,0201	0,02010	0,00000	0,00000
29	0,0301	0,0301	0,0302	0,0302	0,0301	0,0301	0,03014	0,00005	0,00007
30	0,0402	0,0401	0,0400	0,0400	0,0401	0,0400	0,04004	0,00005	0,00007
31	0,0503	0,0502	0,0502	0,0502	0,0502	0,0502	0,05020	0,00000	0,00000
32	0,0601	0,0600	0,0600	0,0600	0,0600	0,0600	0,06000	0,00000	0,00000
33	0,0700	0,0700	0,0700	0,0700	0,0700	0,0700	0,07000	0,00000	0,00000
34	0,0801	0,0800	0,0801	0,0801	0,0800	0,0800	0,08004	0,00005	0,00007
35	0,0902	0,0899	0,0899	0,0899	0,0899	0,0899	0,08990	0,00000	0,00000
36	0,1001	0,0999	0,0999	0,0999	0,0999	0,0999	0,09990	0,00000	0,00000
37	0,1100	0,1098	0,1098	0,1098	0,1098	0,1098	0,10980	0,00000	0,00000
38	0,1200	0,1199	0,1198	0,1198	0,1199	0,1198	0,11984	0,00005	0,00007
39	0,1300	0,1299	0,1298	0,1299	0,1299	0,1299	0,12988	0,00004	0,00006
40	0,1400	0,1398	0,1398	0,1398	0,1398	0,1398	0,13980	0,00000	0,00000
41	0,1500	0,1500	0,1500	0,1500	0,1500	0,1500	0,15000	0,00000	0,00000
42	0,1600	0,1599	0,1599	0,1599	0,1599	0,1599	0,15990	0,00000	0,00000
43	0,1699	0,1698	0,1697	0,1698	0,1698	0,1697	0,16976	0,00005	0,00007
44	0,1799	0,1798	0,1799	0,1799	0,1799	0,1799	0,17988	0,00004	0,00006
45	0,1900	0,1897	0,1897	0,1897	0,1897	0,1897	0,18970	0,00000	0,00000
46	0,2000	0,1998	0,1998	0,1998	0,1998	0,1998	0,19980	0,00000	0,00000
47	0,2100	0,2098	0,2098	0,2098	0,2098	0,2098	0,20980	0,00000	0,00000
48	0,2200	0,2197	0,2197	0,2197	0,2197	0,2197	0,21970	0,00000	0,00000
49	0,2300	0,2298	0,2298	0,2298	0,2298	0,2298	0,22980	0,00000	0,00000
50	0,2401	0,2397	0,2397	0,2397	0,2397	0,2397	0,23970	0,00000	0,00000
51	0,2501	0,2499	0,2499	0,2499	0,2499	0,2499	0,24990	0,00000	0,00000
								σ_R	δ_{max}
								0,000020	0,00007

Tabela 5-40: Medidas obtidas com $K_{0,200}$ calculado para D = 0,200mm.

T	d_{REF}	d_1	d_2	d_3	d_4	d_5	d	s	δ
01	-0,2492	-0,2496	-0,2496	-0,2495	-0,2496	-0,2496	-0,24958	0,00004	0,00006
02	-0,2396	-0,2399	-0,2400	-0,2400	-0,2400	-0,2399	-0,23996	0,00005	0,00007
03	-0,2300	-0,2301	-0,2301	-0,2301	-0,2301	-0,2301	-0,23010	0,00000	0,00000
04	-0,2200	-0,2201	-0,2202	-0,2202	-0,2202	-0,2202	-0,22018	0,00004	0,00006
05	-0,2100	-0,2103	-0,2102	-0,2102	-0,2102	-0,2102	-0,21022	0,00004	0,00006
06	-0,2001	-0,2002	-0,2002	-0,2002	-0,2002	-0,2002	-0,20020	0,00000	0,00000
07	-0,1901	-0,1902	-0,1902	-0,1902	-0,1902	-0,1902	-0,19020	0,00000	0,00000
08	-0,1801	-0,1802	-0,1802	-0,1801	-0,1801	-0,1802	-0,18016	0,00005	0,00007
09	-0,1702	-0,1701	-0,1701	-0,1701	-0,1701	-0,1702	-0,17012	0,00004	0,00006
10	-0,1602	-0,1602	-0,1602	-0,1602	-0,1602	-0,1602	-0,16020	0,00000	0,00000
11	-0,1501	-0,1500	-0,1500	-0,1500	-0,1500	-0,1500	-0,15000	0,00000	0,00000
12	-0,1402	-0,1400	-0,1401	-0,1401	-0,1401	-0,1401	-0,14008	0,00004	0,00006
13	-0,1302	-0,1301	-0,1301	-0,1301	-0,1301	-0,1301	-0,13010	0,00000	0,00000
14	-0,1202	-0,1200	-0,1200	-0,1200	-0,1200	-0,1200	-0,12000	0,00000	0,00000
15	-0,1102	-0,1100	-0,1101	-0,1101	-0,1101	-0,1101	-0,11008	0,00004	0,00006
16	-0,1002	-0,0999	-0,0999	-0,1000	-0,1000	-0,1000	-0,09996	0,00005	0,00007
17	-0,0903	-0,0899	-0,0900	-0,0900	-0,0900	-0,0900	-0,08998	0,00004	0,00006
18	-0,0802	-0,0800	-0,0801	-0,0800	-0,0800	-0,0800	-0,08002	0,00004	0,00006
19	-0,0702	-0,0700	-0,0699	-0,0699	-0,0699	-0,0699	-0,06992	0,00004	0,00006
20	-0,0601	-0,0599	-0,0599	-0,0599	-0,0599	-0,0600	-0,05992	0,00004	0,00006
21	-0,0500	-0,0498	-0,0498	-0,0498	-0,0498	-0,0498	-0,04980	0,00000	0,00000
22	-0,0400	-0,0399	-0,0398	-0,0398	-0,0398	-0,0399	-0,03984	0,00005	0,00007
23	-0,0300	-0,0299	-0,0299	-0,0299	-0,0299	-0,0299	-0,02990	0,00000	0,00000
24	-0,0200	-0,0199	-0,0199	-0,0199	-0,0199	-0,0198	-0,01988	0,00004	0,00006
25	-0,0100	-0,0099	-0,0099	-0,0099	-0,0099	-0,0099	-0,00990	0,00000	0,00000
26	0,0002	0,0001	0,0002	0,0002	0,0002	0,0002	0,00018	0,00004	0,00006
27	0,0101	0,0101	0,0101	0,0102	0,0102	0,0102	0,01016	0,00005	0,00007
28	0,0201	0,0201	0,0201	0,0201	0,0201	0,0201	0,02010	0,00000	0,00000
29	0,0301	0,0301	0,0301	0,0301	0,0301	0,0301	0,03010	0,00000	0,00000
30	0,0402	0,0401	0,0401	0,0401	0,0402	0,0401	0,04012	0,00004	0,00006
31	0,0503	0,0502	0,0502	0,0502	0,0502	0,0502	0,05020	0,00000	0,00000
32	0,0601	0,0600	0,0601	0,0601	0,0600	0,0601	0,06006	0,00005	0,00007
33	0,0700	0,0701	0,0701	0,0701	0,0701	0,0701	0,07010	0,00000	0,00000
34	0,0801	0,0801	0,0800	0,0800	0,0800	0,0800	0,08002	0,00004	0,00006
35	0,0902	0,0900	0,0900	0,0901	0,0901	0,0901	0,09006	0,00005	0,00007
36	0,1001	0,1000	0,1000	0,1000	0,1000	0,1000	0,10000	0,00000	0,00000
37	0,1100	0,1099	0,1099	0,1099	0,1099	0,1099	0,10990	0,00000	0,00000
38	0,1200	0,1199	0,1200	0,1199	0,1199	0,1199	0,11992	0,00004	0,00006
39	0,1300	0,1300	0,1300	0,1299	0,1299	0,1299	0,12994	0,00005	0,00007
40	0,1400	0,1399	0,1399	0,1400	0,1399	0,1400	0,13994	0,00005	0,00007
41	0,1500	0,1501	0,1501	0,1501	0,1501	0,1501	0,15010	0,00000	0,00000
42	0,1600	0,1600	0,1599	0,1599	0,1599	0,1599	0,15992	0,00004	0,00006
43	0,1699	0,1699	0,1699	0,1699	0,1699	0,1700	0,16992	0,00004	0,00006
44	0,1799	0,1800	0,1800	0,1800	0,1800	0,1800	0,18000	0,00000	0,00000
45	0,1900	0,1898	0,1898	0,1898	0,1898	0,1899	0,18982	0,00004	0,00006
46	0,2000	0,1999	0,1999	0,1999	0,1999	0,1999	0,19990	0,00000	0,00000
47	0,2100	0,2100	0,2100	0,2099	0,2099	0,2099	0,20994	0,00005	0,00007
48	0,2200	0,2199	0,2199	0,2199	0,2199	0,2199	0,21990	0,00000	0,00000
49	0,2300	0,2300	0,2300	0,2300	0,2300	0,2300	0,23000	0,00000	0,00000
50	0,2401	0,2399	0,2399	0,2399	0,2399	0,2399	0,23990	0,00000	0,00000
51	0,2501	0,2501	0,2501	0,2501	0,2501	0,2501	0,25010	0,00000	0,00000
								σ_R	δ_{max}
								0,000027	0,00007

Tabela 5-41: Medidas obtidas com $K_{0,300}$ calculado para D = 0,300mm.

T	d_{REF}	d_1	d_2	d_3	d_4	d_5	d	s	δ
01	-0,2492	-0,2495	-0,2495	-0,2495	-0,2495	-0,2495	-0,24950	0,00000	0,00000
02	-0,2396	-0,2399	-0,2399	-0,2399	-0,2399	-0,2399	-0,23990	0,00000	0,00000
03	-0,2300	-0,2300	-0,2300	-0,2300	-0,2300	-0,2300	-0,23000	0,00000	0,00000
04	-0,2200	-0,2201	-0,2201	-0,2201	-0,2201	-0,2201	-0,22010	0,00000	0,00000
05	-0,2100	-0,2101	-0,2101	-0,2101	-0,2101	-0,2101	-0,21010	0,00000	0,00000
06	-0,2001	-0,2001	-0,2001	-0,2001	-0,2001	-0,2001	-0,20010	0,00000	0,00000
07	-0,1901	-0,1901	-0,1901	-0,1901	-0,1901	-0,1901	-0,19010	0,00000	0,00000
08	-0,1801	-0,1801	-0,1801	-0,1800	-0,1801	-0,1801	-0,18008	0,00004	0,00006
09	-0,1702	-0,1701	-0,1701	-0,1701	-0,1701	-0,1701	-0,17010	0,00000	0,00000
10	-0,1602	-0,1601	-0,1601	-0,1601	-0,1601	-0,1601	-0,16010	0,00000	0,00000
11	-0,1501	-0,1500	-0,1500	-0,1500	-0,1500	-0,1500	-0,15000	0,00000	0,00000
12	-0,1402	-0,1401	-0,1401	-0,1401	-0,1401	-0,1401	-0,14010	0,00000	0,00000
13	-0,1302	-0,1300	-0,1300	-0,1300	-0,1300	-0,1300	-0,13000	0,00000	0,00000
14	-0,1202	-0,1199	-0,1200	-0,1199	-0,1199	-0,1199	-0,11992	0,00004	0,00006
15	-0,1102	-0,1100	-0,1100	-0,1101	-0,1100	-0,1100	-0,11002	0,00004	0,00006
16	-0,1002	-0,0999	-0,0999	-0,0999	-0,0999	-0,0999	-0,09990	0,00000	0,00000
17	-0,0903	-0,0899	-0,0900	-0,0899	-0,0899	-0,0899	-0,08992	0,00004	0,00006
18	-0,0802	-0,0800	-0,0800	-0,0800	-0,0800	-0,0800	-0,08000	0,00000	0,00000
19	-0,0702	-0,0699	-0,0699	-0,0698	-0,0698	-0,0699	-0,06986	0,00005	0,00007
20	-0,0601	-0,0599	-0,0599	-0,0599	-0,0599	-0,0599	-0,05990	0,00000	0,00000
21	-0,0500	-0,0498	-0,0498	-0,0498	-0,0498	-0,0498	-0,04980	0,00000	0,00000
22	-0,0400	-0,0398	-0,0399	-0,0398	-0,0398	-0,0398	-0,03982	0,00004	0,00006
23	-0,0300	-0,0299	-0,0299	-0,0299	-0,0299	-0,0299	-0,02990	0,00000	0,00000
24	-0,0200	-0,0199	-0,0198	-0,0198	-0,0198	-0,0199	-0,01984	0,00005	0,00007
25	-0,0100	-0,0099	-0,0099	-0,0099	-0,0099	-0,0099	-0,00990	0,00000	0,00000
26	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,00020	0,00000	0,00000
27	0,0101	0,0102	0,0102	0,0102	0,0102	0,0102	0,01020	0,00000	0,00000
28	0,0201	0,0201	0,0201	0,0201	0,0201	0,0201	0,02010	0,00000	0,00000
29	0,0301	0,0301	0,0301	0,0301	0,0301	0,0301	0,03010	0,00000	0,00000
30	0,0402	0,0401	0,0401	0,0401	0,0401	0,0401	0,04010	0,00000	0,00000
31	0,0503	0,0502	0,0502	0,0502	0,0502	0,0502	0,05020	0,00000	0,00000
32	0,0601	0,0600	0,0601	0,0601	0,0600	0,0600	0,06004	0,00005	0,00007
33	0,0700	0,0701	0,0700	0,0701	0,0701	0,0700	0,07006	0,00005	0,00007
34	0,0801	0,0800	0,0800	0,0800	0,0800	0,0800	0,08000	0,00000	0,00000
35	0,0902	0,0900	0,0900	0,0900	0,0900	0,0900	0,09000	0,00000	0,00000
36	0,1001	0,1000	0,1000	0,1000	0,1000	0,1000	0,10000	0,00000	0,00000
37	0,1100	0,1098	0,1099	0,1099	0,1099	0,1098	0,10986	0,00005	0,00007
38	0,1200	0,1199	0,1198	0,1199	0,1199	0,1199	0,11988	0,00004	0,00006
39	0,1300	0,1299	0,1298	0,1299	0,1299	0,1299	0,12988	0,00004	0,00006
40	0,1400	0,1399	0,1399	0,1399	0,1399	0,1399	0,13990	0,00000	0,00000
41	0,1500	0,1500	0,1500	0,1500	0,1500	0,1500	0,15000	0,00000	0,00000
42	0,1600	0,1599	0,1598	0,1599	0,1599	0,1599	0,15988	0,00004	0,00006
43	0,1699	0,1699	0,1699	0,1699	0,1699	0,1699	0,16990	0,00000	0,00000
44	0,1799	0,1799	0,1799	0,1799	0,1799	0,1799	0,17990	0,00000	0,00000
45	0,1900	0,1898	0,1898	0,1898	0,1898	0,1898	0,18980	0,00000	0,00000
46	0,2000	0,1998	0,1998	0,1998	0,1998	0,1998	0,19980	0,00000	0,00000
47	0,2100	0,2099	0,2098	0,2098	0,2098	0,2099	0,20984	0,00005	0,00007
48	0,2200	0,2198	0,2199	0,2198	0,2198	0,2198	0,21982	0,00004	0,00006
49	0,2300	0,2299	0,2299	0,2299	0,2299	0,2299	0,22990	0,00000	0,00000
50	0,2401	0,2398	0,2398	0,2398	0,2398	0,2398	0,23980	0,00000	0,00000
51	0,2501	0,2500	0,2500	0,2500	0,2500	0,2500	0,25000	0,00000	0,00000
								σ_R	δ_{max}
								0,000014	0,00007

Tabela 5-42: Medidas obtidas com $K_{0,400}$ calculado para D = 0,400mm.

Sejam ρ_i e s, respectivamente, o valor médio e o desvio padrão estimados para a posição $\hat{\rho}_i$ de uma dada reta $\mathcal{R}_i$, medida em *nº de pixels* com relação ao centro de gravidade da distribuição de retas e seja $\hat{\rho}_i'$ a posição dessa mesma reta após translação da imagem de uma distância $\hat{D}$ com média $\overline{D}$ (em mm) e desvio padrão σ_D. Portanto, a translação $\hat{t}_i$ da reta $\mathcal{R}_i$ apresenta média $\bar{t}_i = \left|\rho_i - \rho_i'\right|$ e desvio padrão $s\sqrt{2}$ e o conjunto das retas da imagem desloca-se de uma distância $\hat{t}$ com média $\sum \bar{t}_i / n$ e desvio padrão $s\sqrt{2/n}$ (figura 5-14).

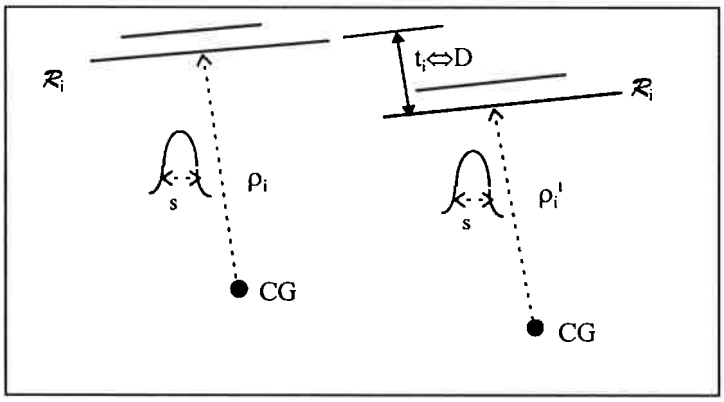

Figura 5.14: Incerteza no espaçamento das retas da imagem.

Nessas condições, o fator de escala $\hat{k}$ (em *nº de pixels*/mm) é, simplesmente:

$$\hat{k} = \hat{t}/\hat{D} \tag{5-8}$$

Linearizando-se essa expressão em torno do ponto $(\bar{t}, \overline{D})$, resulta:

$$\hat{k}(\bar{t}, \overline{D}) \cong \frac{\bar{t}}{\overline{D}} + \frac{\hat{t} - \bar{t}}{\overline{D}} - \frac{\hat{D} - \overline{D}}{\overline{D}^2}\bar{t} \tag{5-9}$$

que é uma variável aleatória de média $\bar{t}/\overline{D}$ e desvio padrão

$$\sigma_k = \sqrt{\frac{2s^2}{n\overline{D}^2} + \frac{\bar{t}^2\sigma_D^2}{\overline{D}^4}} \tag{5-10}$$

Adotando procedimento similar para a expressão das posições $\hat{r}_i$ das retas, medidas em mm, ou seja,

$$\hat{r}_i = \frac{\hat{\rho}_i}{\hat{k}} \tag{5-11}$$

resulta:

$$\hat{r}_i(\bar{\rho}_i, \bar{k}) \cong \frac{\bar{\rho}_i}{\bar{k}} + \frac{\hat{\rho}_i - \bar{\rho}_i}{\bar{k}} - \frac{\hat{k} - \bar{k}}{\bar{k}^2}\bar{\rho}_i \tag{5-12}$$

que é uma variável aleatória de média $\bar{\rho}_i/\bar{k}$ e desvio-padrão

$$\sigma_r^i = \sqrt{\frac{s^2}{\bar{k}^2} + \frac{\bar{\rho}_i^2}{\bar{k}^4}\sigma_k^2} \tag{5-13}$$

Substituindo-se 5-10 em 5-13, obtém-se

$$\sigma_r^i = \frac{1}{\bar{k}}\sqrt{s^2 + \frac{\bar{r}_i^2}{\bar{D}^2}\left(\frac{2s^2}{n} + \bar{k}^2\sigma_D^2\right)} \tag{5-14}$$

expressão que permite avaliar a influência do deslocamento D sobre a incerteza das medidas (em mm) de uma dada reta $\mathcal{R}_i$ da imagem.

Na expressão 5-14, os seguintes parâmetros são conhecidos: 1º) $s \approx 0{,}012$ pixels, conforme estimativa apresentada na tabela 5-38; 2º) $\bar{k} \approx 1000$, ou seja, 1 pixel/μm; 3º) $\sigma_D \approx 0{,}02$μm, de acordo com (LINK, 1997), ou, alternativamente, $\sigma_D \approx 0{,}01$μm, já que aquela referência adota um método de cálculo muito conservador para essa incerteza.

Substituindo-se esses valores em 5-14 e, para efeito de análise comparativa, arbitrando-se uma particular posição r_i (no caso, r_i=0,125mm) para o cálculo de σ_r, obtém-se:

$$\sigma_r(0{,}125) = 10^{-3}\sqrt{1{,}44 \times 10^{-4} + \frac{0{,}0156}{\bar{D}^2}\left(\frac{2{,}88 \times 10^{-4}}{n} + \sigma_D^2\right)} \tag{5-15}$$

expressão que, avaliada para estimativas $\sigma_D \in \{0,01; 0,02\}$ e deslocamentos $D_i \in \{0,10, 0,20, 0,30, 0,40$ e $0,50\}$ correspondendo, respectivamente, a grupos de n retas $\in \{40, 30, 20, 10$ e $1\}$, produz a estimativa teórica da variação da repetibilidade das medidas $\hat{r}_i$ ($\bar{r}_i = 0,125$) em função do deslocamento.

Confrontando-se a variação dos valores médios de repetibilidade reportados nas tabelas 5-39 a 5-42 com o gráfico da figura 5-15, nota-se que, com exceção da condição D=0,300mm, que aparentemente apresenta um erro experimental, os demais valores de σ_R encontram-se dentro do intervalo teórico de variação previsto admitindo-se $\sigma_D = 0,01$ μm. Notando-se ainda, na figura 5-15, que as duas curvas apresentam um ponto de mínimo ao redor de 0,400mm, esse valor será adotado como o deslocamento ideal para a medição de microescalas segundo o método proposto.

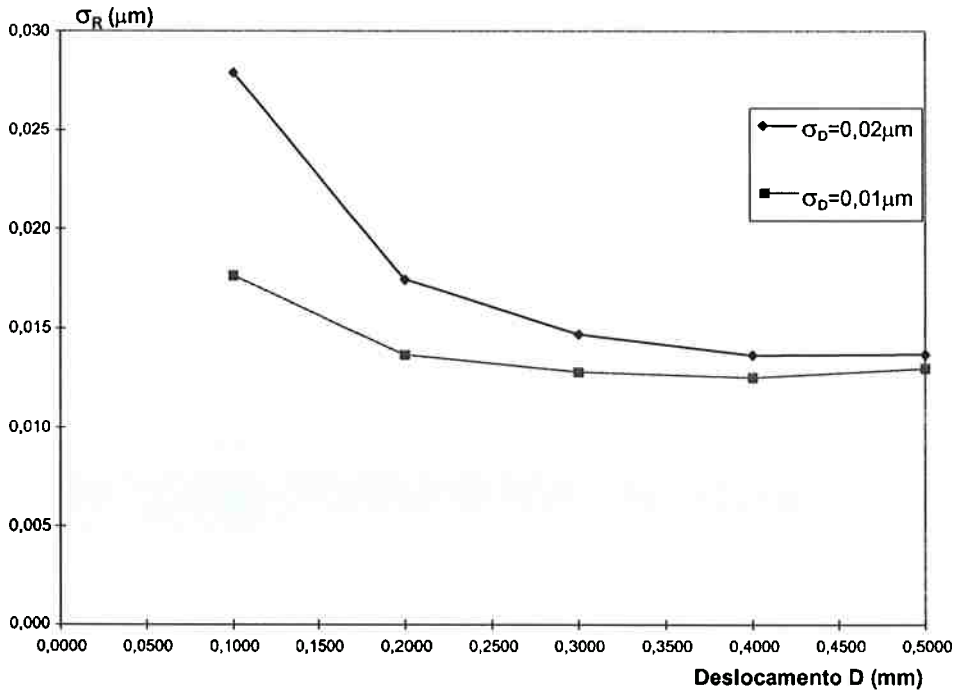

Figura 5-15: Variação da repetibilidade do método em função do deslocamento D.

Na tabela 5-43 apresentam-se as medidas dos 101 traços da microescala, obtidas através do IMVC-E_4 aplicado às duas séries de imagens (traços 1-51 e traços 51-101), utilizando o mesmo fator de escala $K_{0,400}$ baseado no deslocamento de 0,400mm.

T	d_1	d_2	d_3	d_4	d_5	d	s	δ	T	d_1	d_2	d_3	d_4	d_5	d	s	δ
01	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,00000	0,00000	51	0,4995	0,4995	0,4995	0,4995	0,4995	0,4995	0,00000	0,00000
02	0,0096	0,0096	0,0096	0,0096	0,0096	0,0096	0,00000	0,00000	52	0,5094	0,5093	0,5093	0,5094	0,5094	0,5093	0,00005	0,00007
03	0,0195	0,0195	0,0195	0,0195	0,0195	0,0195	0,00000	0,00000	53	0,5194	0,5193	0,5193	0,5194	0,5193	0,5193	0,00005	0,00007
04	0,0294	0,0294	0,0294	0,0294	0,0294	0,0294	0,00000	0,00000	54	0,5294	0,5294	0,5294	0,5294	0,5294	0,5294	0,00000	0,00000
05	0,0394	0,0394	0,0394	0,0394	0,0394	0,0394	0,00000	0,00000	55	0,5394	0,5393	0,5393	0,5394	0,5394	0,5394	0,00005	0,00007
06	0,0494	0,0494	0,0494	0,0494	0,0494	0,0494	0,00000	0,00000	56	0,5494	0,5494	0,5494	0,5495	0,5494	0,5494	0,00004	0,00006
07	0,0594	0,0594	0,0594	0,0594	0,0594	0,0594	0,00000	0,00000	57	0,5593	0,5593	0,5593	0,5593	0,5593	0,5593	0,00000	0,00000
08	0,0694	0,0694	0,0695	0,0694	0,0694	0,0694	0,00004	0,00006	58	0,5693	0,5692	0,5692	0,5694	0,5692	0,5692	0,00009	0,00011
09	0,0794	0,0794	0,0794	0,0794	0,0794	0,0794	0,00000	0,00000	59	0,5793	0,5793	0,5793	0,5794	0,5792	0,5793	0,00007	0,00009
10	0,0894	0,0894	0,0894	0,0894	0,0894	0,0894	0,00000	0,00000	60	0,5893	0,5893	0,5893	0,5893	0,5893	0,5893	0,00000	0,00000
11	0,0995	0,0995	0,0995	0,0995	0,0995	0,0995	0,00000	0,00000	61	0,5994	0,5994	0,5994	0,5995	0,5994	0,5994	0,00004	0,00006
12	0,1094	0,1094	0,1094	0,1094	0,1094	0,1094	0,00000	0,00000	62	0,6093	0,6094	0,6094	0,6094	0,6093	0,6093	0,00005	0,00007
13	0,1195	0,1195	0,1195	0,1195	0,1195	0,1195	0,00000	0,00000	63	0,6193	0,6193	0,6193	0,6194	0,6193	0,6193	0,00004	0,00006
14	0,1296	0,1295	0,1296	0,1296	0,1296	0,1296	0,00004	0,00006	64	0,6293	0,6293	0,6293	0,6294	0,6292	0,6293	0,00007	0,00009
15	0,1395	0,1395	0,1394	0,1395	0,1395	0,1395	0,00004	0,00006	65	0,6392	0,6393	0,6392	0,6393	0,6393	0,6393	0,00005	0,00007
16	0,1496	0,1496	0,1496	0,1496	0,1496	0,1496	0,00000	0,00000	66	0,6493	0,6493	0,6493	0,6494	0,6493	0,6493	0,00004	0,00006
17	0,1596	0,1595	0,1596	0,1596	0,1596	0,1596	0,00004	0,00006	67	0,6594	0,6594	0,6594	0,6595	0,6593	0,6594	0,00007	0,00009
18	0,1695	0,1695	0,1695	0,1695	0,1695	0,1695	0,00000	0,00000	68	0,6693	0,6694	0,6693	0,6694	0,6694	0,6693	0,00005	0,00007
19	0,1796	0,1796	0,1797	0,1797	0,1796	0,1796	0,00005	0,00007	69	0,6794	0,6793	0,6793	0,6795	0,6794	0,6794	0,00008	0,00010
20	0,1896	0,1896	0,1896	0,1896	0,1896	0,1896	0,00000	0,00000	70	0,6893	0,6894	0,6894	0,6895	0,6893	0,6894	0,00008	0,00010
21	0,1997	0,1997	0,1997	0,1997	0,1997	0,1997	0,00000	0,00000	71	0,6996	0,6996	0,6996	0,6997	0,6996	0,6996	0,00004	0,00006
22	0,2097	0,2096	0,2097	0,2097	0,2097	0,2097	0,00004	0,00006	72	0,7095	0,7094	0,7095	0,7096	0,7095	0,7095	0,00007	0,00009
23	0,2196	0,2196	0,2196	0,2196	0,2196	0,2196	0,00000	0,00000	73	0,7194	0,7195	0,7195	0,7196	0,7194	0,7194	0,00008	0,00010
24	0,2296	0,2297	0,2297	0,2297	0,2296	0,2297	0,00005	0,00007	74	0,7295	0,7294	0,7294	0,7295	0,7295	0,7295	0,00005	0,00007
25	0,2396	0,2396	0,2396	0,2396	0,2396	0,2396	0,00000	0,00000	75	0,7395	0,7394	0,7394	0,7396	0,7394	0,7395	0,00009	0,00011
26	0,2497	0,2497	0,2497	0,2497	0,2497	0,2497	0,00000	0,00000	76	0,7496	0,7496	0,7496	0,7497	0,7496	0,7496	0,00004	0,00006
27	0,2597	0,2597	0,2597	0,2597	0,2597	0,2597	0,00000	0,00000	77	0,7596	0,7595	0,7595	0,7596	0,7596	0,7595	0,00005	0,00007
28	0,2696	0,2696	0,2696	0,2696	0,2696	0,2696	0,00000	0,00000	78	0,7695	0,7695	0,7695	0,7697	0,7695	0,7695	0,00009	0,00011
29	0,2796	0,2796	0,2796	0,2796	0,2796	0,2796	0,00000	0,00000	79	0,7795	0,7796	0,7796	0,7796	0,7796	0,7796	0,00004	0,00006
30	0,2896	0,2896	0,2896	0,2896	0,2896	0,2896	0,00000	0,00000	80	0,7895	0,7895	0,7895	0,7896	0,7895	0,7895	0,00004	0,00006
31	0,2997	0,2997	0,2997	0,2997	0,2997	0,2997	0,00000	0,00000	81	0,7997	0,7997	0,7997	0,7998	0,7997	0,7997	0,00004	0,00006
32	0,3095	0,3096	0,3096	0,3095	0,3095	0,3095	0,00005	0,00007	82	0,8095	0,8096	0,8096	0,8096	0,8096	0,8096	0,00004	0,00006
33	0,3196	0,3195	0,3196	0,3196	0,3195	0,3196	0,00005	0,00007	83	0,8196	0,8195	0,8195	0,8197	0,8195	0,8195	0,00009	0,00011
34	0,3295	0,3295	0,3295	0,3295	0,3295	0,3295	0,00000	0,00000	84	0,8295	0,8295	0,8295	0,8296	0,8295	0,8296	0,00004	0,00006
35	0,3395	0,3395	0,3395	0,3395	0,3395	0,3395	0,00000	0,00000	85	0,8395	0,8395	0,8395	0,8396	0,8395	0,8395	0,00004	0,00006
36	0,3495	0,3495	0,3495	0,3495	0,3495	0,3495	0,00000	0,00000	86	0,8495	0,8495	0,8495	0,8497	0,8496	0,8496	0,00009	0,00011
37	0,3593	0,3594	0,3594	0,3594	0,3593	0,3594	0,00005	0,00007	87	0,8594	0,8595	0,8595	0,8596	0,8595	0,8595	0,00007	0,00009
38	0,3694	0,3693	0,3694	0,3694	0,3694	0,3694	0,00004	0,00006	88	0,8695	0,8694	0,8694	0,8696	0,8695	0,8694	0,00008	0,00010
39	0,3794	0,3793	0,3794	0,3794	0,3794	0,3794	0,00004	0,00006	89	0,8794	0,8794	0,8794	0,8795	0,8794	0,8795	0,00004	0,00006
40	0,3894	0,3894	0,3894	0,3894	0,3894	0,3894	0,00000	0,00000	90	0,8893	0,8894	0,8894	0,8894	0,8894	0,8894	0,00004	0,00006
41	0,3995	0,3995	0,3995	0,3995	0,3995	0,3995	0,00000	0,00000	91	0,8995	0,8995	0,8995	0,8996	0,8995	0,8995	0,00004	0,00006
42	0,4094	0,4093	0,4094	0,4094	0,4094	0,4094	0,00004	0,00006	92	0,9093	0,9094	0,9094	0,9095	0,9094	0,9094	0,00007	0,00009
43	0,4194	0,4194	0,4194	0,4194	0,4194	0,4194	0,00000	0,00000	93	0,9193	0,9193	0,9193	0,9194	0,9194	0,9193	0,00005	0,00007
44	0,4294	0,4294	0,4294	0,4294	0,4294	0,4294	0,00000	0,00000	94	0,9293	0,9293	0,9293	0,9294	0,9293	0,9294	0,00004	0,00006
45	0,4393	0,4393	0,4393	0,4393	0,4393	0,4393	0,00000	0,00000	95	0,9392	0,9393	0,9393	0,9394	0,9393	0,9393	0,00007	0,00009
46	0,4493	0,4493	0,4493	0,4493	0,4493	0,4493	0,00000	0,00000	96	0,9493	0,9493	0,9493	0,9494	0,9494	0,9493	0,00005	0,00007
47	0,4594	0,4593	0,4593	0,4593	0,4594	0,4593	0,00005	0,00007	97	0,9593	0,9593	0,9593	0,9594	0,9593	0,9593	0,00004	0,00006
48	0,4693	0,4694	0,4693	0,4693	0,4693	0,4693	0,00004	0,00006	98	0,9693	0,9693	0,9693	0,9694	0,9693	0,9693	0,00004	0,00006
49	0,4794	0,4794	0,4794	0,4794	0,4794	0,4794	0,00000	0,00000	99	0,9793	0,9793	0,9793	0,9794	0,9793	0,9794	0,00004	0,00006
50	0,4893	0,4893	0,4893	0,4893	0,4893	0,4893	0,00000	0,00000	100	0,9893	0,9893	0,9892	0,9894	0,9892	0,9893	0,00008	0,00010
51	0,4995	0,4995	0,4995	0,4995	0,4995	0,4995	0,00000	0,00000	101	0,9994	0,9994	0,9994	0,9995	0,9994	0,9994	0,00004	0,00006
																σ_R	δ_{max}
																0,000035	0,00011

Tabela 5-43: Medidas da microescala completa.

Os resultados anteriores demostram inequivocamente que ambos os processos de medição de microescalas baseados no IMVC-E_4 produzem resultados muito mais precisos que o método convencional. No primeiro processo, a incerteza prevista é da ordem de ±0,04μm, enquanto que, no segundo, é de cerca de ±0,10μm. Ademais, é importante ainda frisar que a incerteza calculada em (LINK, 1997) refere-se a medidas realizadas por um mesmo operador metrologista, de modo que, possivelmente, aquele valor (±0,2μm) seria ainda maior caso os diferentes graus de destreza e vieses visuais tipicamente humanos interferissem na operação do instrumental do ensaio.

Quanto à eficiência dos novos processos desenvolvidos, a diferença, em relação ao método convencional é ainda mais significativa. No Laboratório de Metrologia do IPT, ensaios de calibração de 21 traços de microescalas de 1mm, através de 5 séries de medidas, consomem cerca de 2,5 horas, neste total não estando incluídos o tempo para montagem dos equipamentos (≈30 min) e o tempo para elaboração de planilhas e relatórios (≈30 min); portanto, cerca de 3,5 horas para se calibrar apenas 21 traços. Os novos processos de medição baseados no IMVC-E_4 , por seu turno, calibram sempre o total de traços da microescala (101, portanto 5 vezes mais dados) com uma eficiência até 30 vezes maior.

Utilizando-se 5 séries de medidas, o primeiro processo, baseado em imagens-referência, requer cerca de 30 min para que se realizem as seguintes atividades: 1ª) montagem de equipamentos: ≈5 min; 2ª) captura de 5 imagens-referência e 5 pares de imagens-teste (traços 1-51 e 51-101): ≈10 min; 3ª) cálculo do fator de escala-médio a partir das 5 imagens-referência: ≈5 min[13]. 4ª) medição dos 5 pares de imagens-teste e emissão de relatório: ≈ 10 min. Já o segundo processo requer cerca de 60 min para a

[13] Utilizando-se um computador PC-Pentium com 75 MHz e 32 MB de memória RAM.

execução das seguintes atividades: 1ª) montagem de equipamentos ≈ 30 min; 2ª) captura de 5 pares de imagens dos traços 1-51 deslocadas de 0,400 mm e de 5 imagens dos traços 51-101: ≈15 min; 3ª) cálculo do fator de escala utilizando-se os 5 pares de imagens deslocadas: ≈10 min; 4ª) medição dos 5 pares (1-51 e 51-101) de imagens e emissão de relatório: ≈ 5 min.

5.8 Medição do Desempenho dos Instrumentos Metrológicos para Peneiras

Conforme ilustrado na figura 5.16, o IMVC-P_1 e o IMVC-P_2 medem a escala nominal, respectivamente, através de um detector morfológico de bordas e de um algoritmo de limiarização, razão pela qual em ambos os instrumentos a escolha do limiar tem um papel crucial.

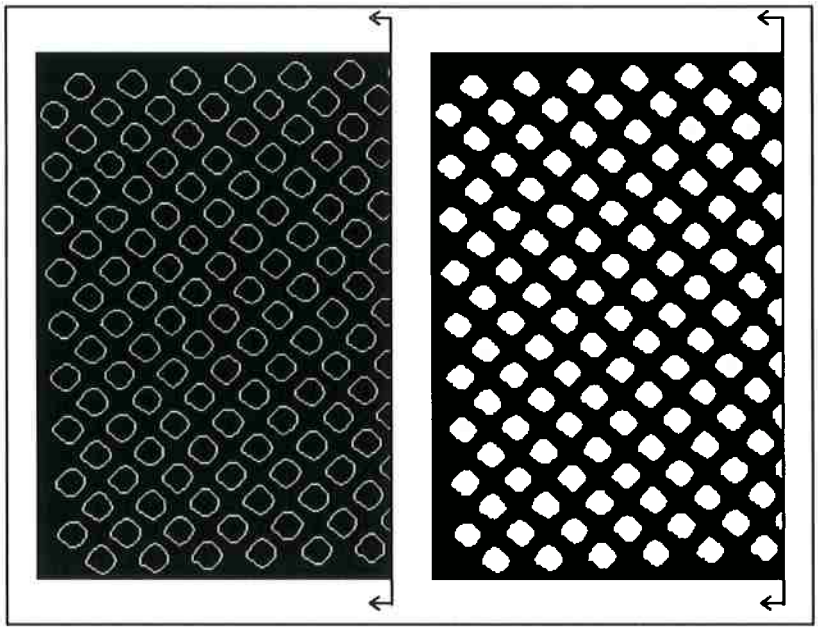

Figura 5.16: Medição em escala nominal: (a) IMVC-P_1 →bordas; (b) IMVC-P2 →regiões.

Para o IMVC-P_2, a imagem deve estar necessariamente limiarizada, sem o que os operadores morfológicos não têm como realizar as medições. Contudo, no caso do IMVC-P_1, o motivo para não se utilizar um detector de bordas convencional está ligado

à necessidade de se eliminar os objetos incompletos junto à moldura da imagem que, de outra forma, introduziriam viés nas medidas. Operando-se com imagens limiarizadas, essa tarefa é facilmente realizada utilizando-se um algoritmo clássico de morfologia matemática (ver equação 3-56), o que torna a aplicação do detector morfológico de bordas uma escolha bastante conveniente.

Em ambos os instrumentos, poder-se-ia introduzir viés na medição da escala nominal caso o limiar fosse escolhido de forma arbitrária ou o método de limiarização automático utilizado operasse com um modelo de histograma de níveis de cinza muito diferente do apresentado pelas imagens em consideração. Todavia, o histograma das imagens de microtexturas é tipicamente bimodal, conforme mostrado na figura 5.17, o que favorece, em particular, a aplicação do método de Otsu. Note-se ainda (figura 5.18) que o limiar determinado automaticamente situa-se em uma faixa onde pequenas variações de seu valor (de até ± 8 tons de cinza) incluem ou excluem pixels com baixa correlação espacial.

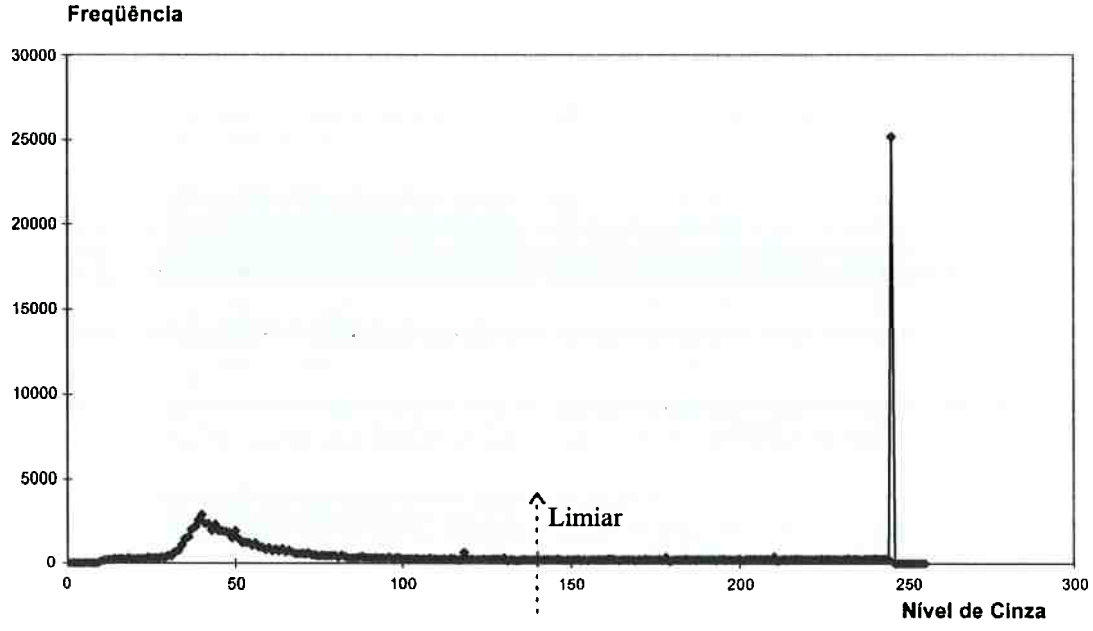

Figura 5.17: Histograma de níveis de cinza de imagem de microtextura de peneira ABNT-325.

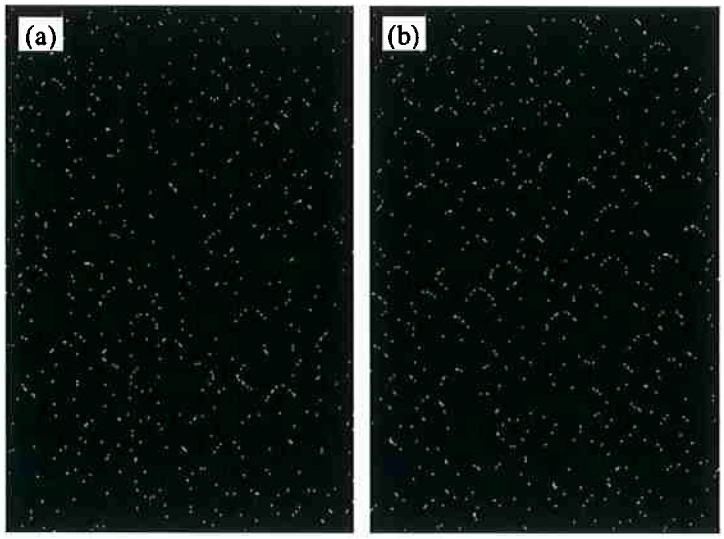

Figura 5.18: Diferença entre duas imagens segmentadas de microtextura: (a) $I(Lim_{OTSU}) - I(Lim_{OTSU} - 8)$; (b) $I(Lim_{OTSU}) - I(Lim_{OTSU} + 8)$.

Com relação à escala ordinal, o IMVC-P_1 utiliza a transformada de Hough para localizar as duas seqüências ordenadas $\mathcal{T}_A$, $\mathcal{T}_B$ de conjuntos de pixels (ver figura 3.34), em torno dos quais retas ortogonais são ajustadas por mínimos quadrados. Para esse instrumento não se realizaram testes sistemáticos para identificação da condição ótima de operação, pois, diferentemente das microescalas, as imagens de microtexturas de peneiras não apresentam um padrão bem definido que possa ser tomado como referência para avaliação do desempenho dos algoritmos. Assim é, que, após alguns experimentos, fixaram-se os seguintes parâmetros para operação da transformada de Hough: θ_{min}=40°, θ_{max}=50°, $d\theta$=0,50°, s_σ=6, L_{min}=200, E=1,0.

Para o IMVC-P_2, as escalas ordinal e racional são medidas através de operadores morfológicos, que produzem uma série ordenada de conjuntos de objetos com dimensões similares (ver figura 5.19). Tais operadores utilizam elementos estruturantes de diâmetro dado (figura 5.20) para inspeção das tolerâncias das aberturas localizadas na imagem, e nenhum outro parâmetro adicional que necessite ser ajustado.

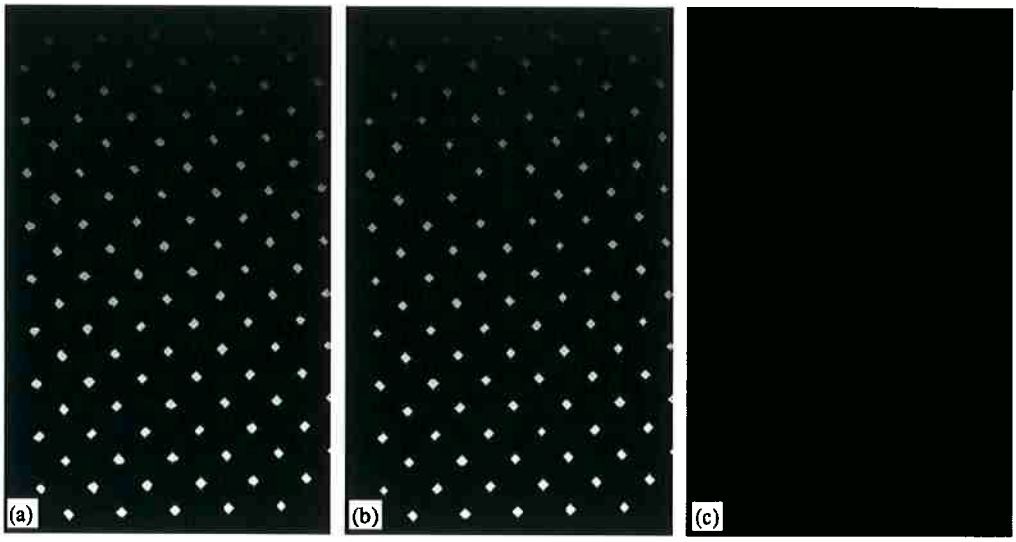

Figura 5.19: Resultado da medição em escala ordinal utilizando o IMVC-P_2: (a) Objetos com raio >6 pixels; (b) Objetos com raio >7 pixels; (c) Objetos com raio >8 pixels.

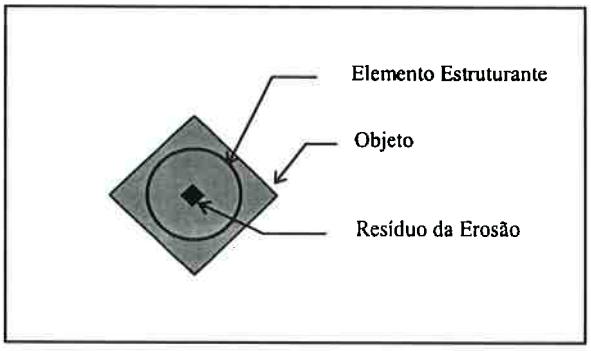

Figura 5.20: Inspeção de tolerância utilizando elemento estruturante Círculo.

Feitas as considerações anteriores, realizaram-se experimentos para avaliação da repetibilidade [14] dos dois instrumentos, a partir de medições baseadas em cinco imagens da microtextura deslocadas de distâncias arbitrárias da ordem de 5 μm em relação a uma posição inicial de referência. Além disso, considerou-se também a influência do limiar escolhido, admitindo-se uma faixa de variação de ±8 níveis de cinza em torno do valor calculado automaticamente.

[14] Não há como avaliar a exatidão relativamente às medidas de referência, pois as amostras utilizadas, em ambos os casos, referem-se a regiões escolhidas aleatoriamente na peneira.

5.8.1 Medição da Repetibilidade do IMVC-P_1

Na tabela 5-44 apresentam-se as estimativas de repetibilidade do IMVC-P_1, onde *a* e *df* são a abertura e o diâmetro de fio médios e s_a, s_d, δ_a e δ_d são os correspondentes valores do desvio-padrão e da incerteza, admitindo-se um grau de significância de 95%.

Imagem	L_{OTSU}-8 a	L_{OTSU} a	L_{OTSU}+8 a		L_{OTSU}-8 df	L_{OTSU} df	L_{OTSU}+8 df
1	45,0	45,0	45,0		34,1	34,1	34,1
2	45,1	45,1	45,1		34,1	34,1	34,1
3	45,0	45,0	45,0		34,0	34,0	34,0
4	45,0	45,0	45,0		34,1	34,1	34,1
5	45,0	45,0	45,0		34,2	34,2	34,2
E{a}	45,0	45,0	45,0	E{df}	34,1	34,1	34,1
s_a	0,04	0,04	0,04	s_d	0,07	0,07	0,07
δ_a	0,06	0,06	0,06	δ_d	0,09	0,09	0,09

(medidas em μm)

Tabela 5-44: Estimativa da repetibilidade do IMVC-P_1.

Analisando-se esses dados, não se constatam variações nas medidas ocasionadas por diferenças de até ± 8 níveis de cinza em relação ao limiar calculado pelo método de Otsu. Analogamente ao caso do IMVC-E_3, a operação de esqueletonização aplicada na etapa final da detecção morfológica de bordas acaba por eliminar o efeito daqueles objetos mostrados na figura 5.18, tornando indistinguíveis as imagens segmentadas.

Conforme apresentado na mesma tabela, a repetibilidade estimada do IMVC-P_1, da ordem de ±0,1μm, é muito superior à do método tradicional, utilizando projetor de perfil, cujas medidas apresentam incerteza de cerca de ±2μm. Note-se, ainda, que a estimação de repetibilidade do novo instrumento se refere a medições de 5 amostras de imagens contendo aproximadamente 180 aberturas e 30 fios, enquanto que no método tradicional são utilizadas apenas 5 medições de uma mesma abertura e de um único fio, medido aproximadamente na mesma região.

5.8.2 Incerteza das Medidas do IMVC-P_2

Como este instrumento realiza medições de carácter local, imagens limiarizadas da mesma cena deslocada de cerca de ± 5 μm em relação a uma posição inicial de referência, por conterem os mesmos objetos [15], são completamente indistinguíveis aos operadores morfológicos. Por esse motivo, investigou-se, neste caso, apenas o efeito de possível influência do valor do limiar, admitindo-se variações de ± 8 níveis de cinza em relação ao valor calculado pelo método de Otsu.

Na tabela 5-45 mostra-se que variações do limiar redistribuem, de fato, os objetos da imagem entre as várias faixas de tolerância, o que, aliás, era esperado, já que neste caso não se pode contar com o efeito benéfico do processo de esqueletonização referido acima. Entretanto, as alterações observadas são inferiores a 2%, ao nível de 95% de significância. Note-se, ainda, que não há como se comparar esse resultado com algum valor de referência, pois o ensaio tradicional [16] não inclui a verificação de tolerâncias por ser isto totalmente inviável do ponto de vista logístico.

	Faixas de Tolerância (μm)				
Limiar	a < 41	41 < a < 48	48 < a < 58	58 < a < 67	a > 67
130	0,9	99,1	0	0	0
138	1,7	98,3	0	0	0
146	2,2	97,8	0	0	0
s	0,66	0,66	0	0	0
δ	1,63	1,63	0	0	0

(medidas em %)

Tabela 5-45: Incerteza nas medições realizadas pelo IMVC-P_2.

[15] Em tal situação, os objetos que adentram ou saem da cena são automaticamente eliminados pelo algoritmo de eliminação de objetos incompletos.

[16] No caso, trata-se do ensaio realizado no Laboratório de Metrologia do IPT.

5.9 Medições Rotineiras de Microtexturas de Peneiras

Utilizando-se concomitantemente o IMVC-P_1 e o IMVC-P_2, determinam-se os valores médios da abertura da trama e do diâmetro do fio, e ainda se inspecionam as tolerâncias das aberturas segundo intervalos de variação tão estreitos quanto 3μm [17]. Para tanto, é necessário que se determine previamente o fator de escala *K*, convertendo *nº de pixels* em mm, o que, diferentemente do caso da microescala, é mais fácil de se realizar através do método de deslocamento mecânico do que mediante a utilização de algum objeto-referência; como a montagem da peneira na platina do microscópio requer a movimentação do canhão, seria necessário realizar um ajuste fino da distância focal quando da substituição do objeto-referência pela peneira.

Na figura 5.21 apresenta-se o par de imagens correspondentes a uma região da microtextura, deslocada de uma distância *D* conhecida. Note-se ainda que, para identificar retas correspondentes em ambas as imagens, é necessário criar uma fronteira artificial na microtextura, o que, no caso em consideração, foi realizado com auxílio de uma fita adesiva adequadamente colada ao longo de uma das direções da trama.

Para ilustrar a operação dos novos instrumentos de calibração, utilizaram-se inicialmente 5 pares de imagens deslocadas de uma distância *D*=0,500mm, medida com o micrômetro do microscópio, determinando-se o fator de escala *K* a partir da equação 3-48. Em seguida, atendendo à recomendação da norma ABNT-5734 para peneiras com abertura nominal de 45μm, realizaram-se medições e inspeções de tolerância em 14 regiões da microtextura arbitrariamente selecionadas.

[17] O que, para a constante de ampliação geométrica utilizada, corresponde aproximadamente a 1 pixel.

Na tabela 5-46 apresentam-se os valores médios da abertura e do diâmetro do fio obtidos através do IMVC-P_1 e na tabela 5-47 os resultados da inspeção de tolerância realizada pelo IMVC-P_2. Para este último caso, classificaram-se as aberturas a_i em 5 intervalos de variação [18], baseados no valor nominal a_N e nas três tolerâncias X, Y, Z [19] definidas na norma ABNT-5734, que estipula que: "1º) nenhuma abertura a_i deve ser superior a $a_N + X$; 2º) a abertura média $\bar{a}$ deve estar no intervalo $[a_N-Y, a_N+Y]$; 3º) no máximo 6% das aberturas podem estar no intervalo $[a_N+Z, a_N+X]$".

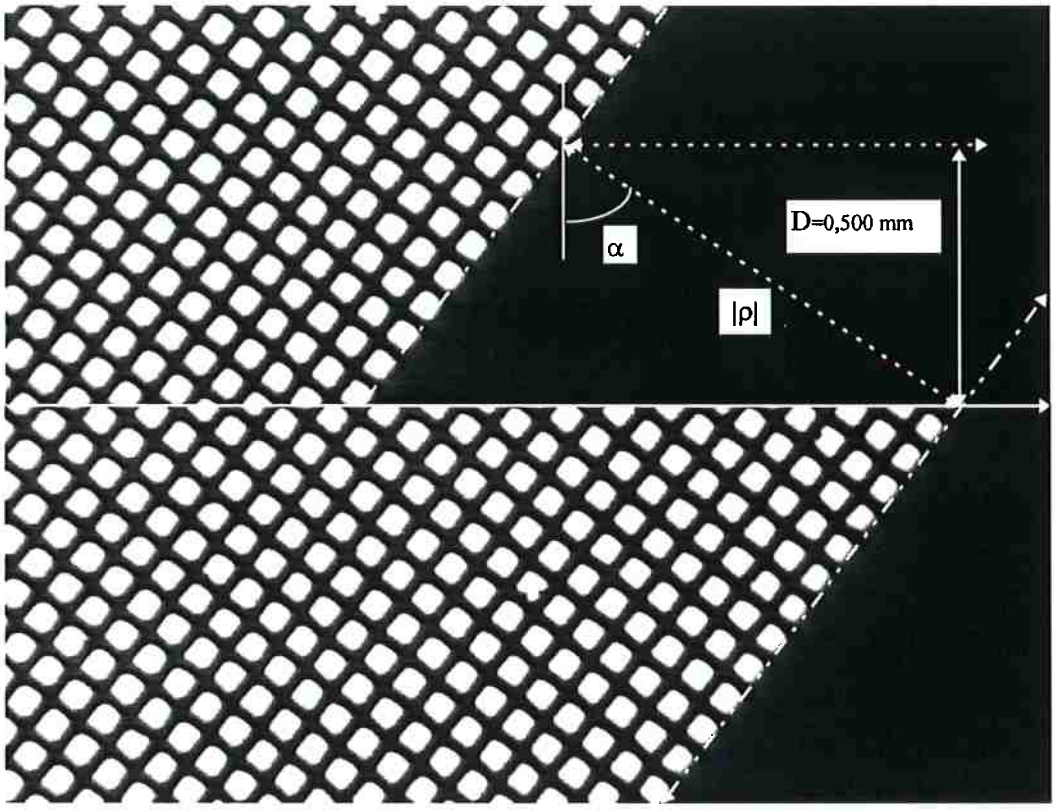

Figura 5-21: Determinação do fator de escala através de deslocamento mecânico da peneira.

[18] A rigor, poderiam ser utilizados muito mais intervalos, já que a resolução do instrumento é da ordem de 3 μm.

[19] Os intervalos, são: $a_i < a_N-Y$, $a_N-Y \leq a_i \leq a_N+Y$, $a_N+Y \leq a_i \leq a_N+Z$, $a_N+Z \leq a_i \leq a_N+X$ e $a_i \geq a_N+X$.

Região	1	2	3	4	5	6	7	8	9	10	11	12	13	13	Média	Admissível
a	45,5	44,1	44,0	44,4	45,1	43,8	43,7	45,0	44,7	43,5	44,8	43,9	44,2	44,1	44,4	40,9 ≤ a ≤ 47,1
df	34,8	34,1	34,2	34,0	34,5	34,1	33,8	34,5	34,7	34,2	33,7	34,6	33,9	34,4	34,2	27,0 ≤ df ≤ 37,0

(medidas em μm)

Tabela 5-46: Medição de 14 regiões da peneira utilizando o IMVC-P_1.

Região	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Média	Admissível
$a_i < 40,9$	0,9	1,3	0,8	3,1	0,5	0,5	0,0	2,6	0,0	0,9	0,0	0,0	0,0	0,4	0,8	—
$40,9 \leq a_i \leq 47,1$	99,1	98,7	99,2	96,9	99,5	99,5	100,0	97,4	100,0	99,1	100,0	100,0	100,0	99,6	99,2	—
$47,1 \leq a_i \leq 57,0$	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	—
$57,0 \leq a_i \leq 66,0$	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	6
$a_i > 66,0$	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0

(medidas em %)

Tabela 5-47: Inspeção de tolerâncias em 14 regiões da peneira utilizando o IMVC-P_2.

Os resultados da medição apresentados acima revelam que a amostra de peneira ABNT-325 satisfaz aos requisitos dimensionais estabelecidos na norma ABNT-5734. Cabe, porém, destacar, que apesar de a inspeção de tolerâncias implementada pelo IMVC-P_2 ter resolução para identificar diferenças em diâmetro de 3 μm (1 pixel, na imagem), não é capaz de diferenciar as deformações reais da abertura daquelas causadas pela presença de corpos estranhos aderidos à trama, o que se evidenciou em muitas das classificações de objetos no intervalo $a_i < 40,9$ μm que, como se pode observar na figura 5.22, não são deformações verdadeiras.

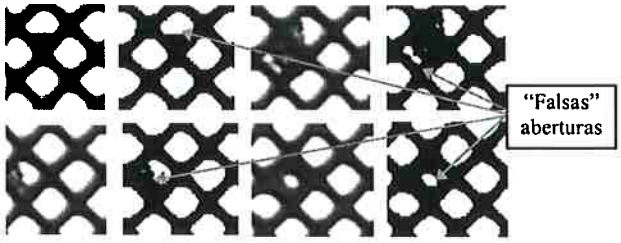

Figura 5.22: Aberturas obstruídas por corpos estranhos.

Conforme já mencionado, o método de calibração de peneiras utilizado pelo Laboratório de Metrologia do IPT baseia-se na norma ASTM E11-87 (ASTM, 1987), que estabelece um número mínimo de duas regiões de observação da microtextura, escolhidas aleatoriamente, onde então se efetuam 10 medições, em ambas as direções, das aberturas a_i e diâmetros df_i, determinando-se os seus valores médios e verificando-se, simultaneamente, se se encontram dentro dos intervalos de tolerância admissíveis (figura 5.23).

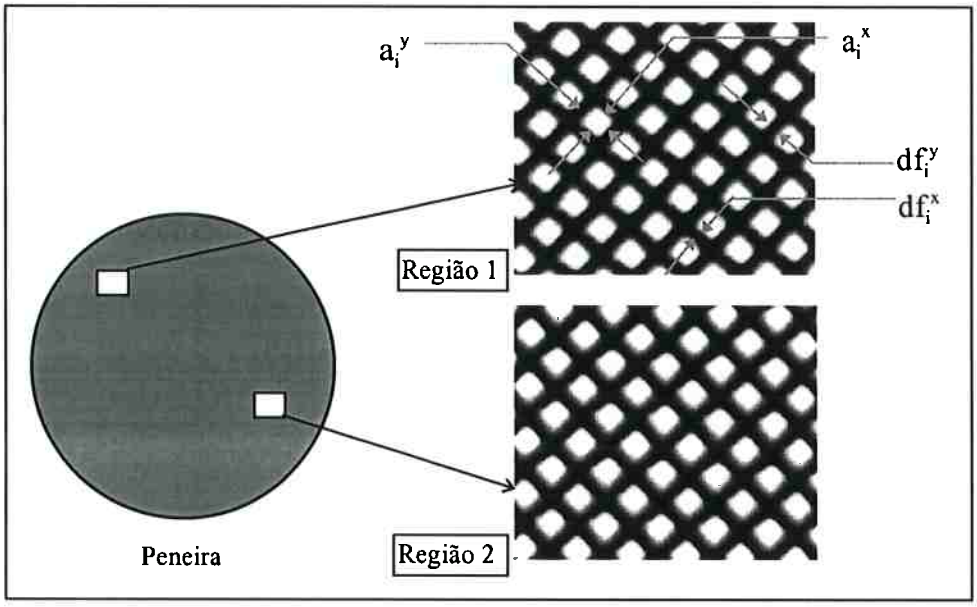

Figura 5.23: Esquema de medição de peneiras segundo a norma ASTM E11-87.

O método referido acima produz uma amostragem insignificante de valores a_i e df_i, o que torna discutível a representatividade das medidas realizadas. Além disso, a calibração de peneiras nessas condições consome cerca de 60 minutos [20], ou seja, 3/4 de minuto para cada medição do par (a_i, df_i) segundo as duas direções ortogonais da microtextura. Todavia, para se melhorar a qualidade dos resultados, adotando-se, por exemplo, o procedimento descrito na norma ABNT-5734, o tempo necessário para a

execução do ensaio seria extremamente elevado, tornando-o inviável do ponto de vista econômico [21].

Em contraste, os dois instrumentos implementados para medição de peneiras permitem, em tempo razoável, operar com amostras significativas da microtextura. Tipicamente, ensaios baseados no uso concomitante desses dois instrumentos, envolvem as seguintes atividades: 1ª) montagem de equipamentos: ≈ 10 min; 2ª) captura de 5 pares de imagens deslocadas de uma distância fixa: ≈ 5 min; 3ª) captura de 14 imagens de regiões distintas da peneira: ≈ 5 min; 4ª) cálculo do fator de escala K: ≈ 5 min; 5ª) determinação dos valores médios da abertura e do diâmetro do fio: ≈ 20 min; 6ª) inspeção de tolerâncias segundo a norma ABNT-5734: ≈ 15 min.

Dessa forma, a medição de 14 regiões distintas da peneira, correspondendo a cerca de 4000 aberturas e respectivos fios adjacentes, pode ser realizada, com os dois instrumentos propostos, em menos de 1 hora. Este ganho é bastante significativo, principalmente se se observar que os novos métodos de medição utilizam amostras 200 vezes maiores que o estipulado na norma ASTM E11-87 e quase 13 vezes maiores do que o requerido pela norma ABNT-5734.

---

[20] Dado fornecido pelo Laboratório de Metrologia do IPT.

[21] Por exemplo, para peneiras padrão ABNT-325 é necessário medir pelo menos 25 aberturas e diâmetros de fios em direções ortogonais em pelo menos 14 áreas distintas de observação, totalizando cerca de 700 medidas e requerendo aproximadamente 9 horas ininterruptas de trabalho !

6 - CONCLUSÕES E RECOMENDAÇÕES

Aplicações de visão computacional em laboratórios metrológicos de padrões secundários são pouco freqüentes, dado que estes operam nos domínios da exatidão e precisão, enquanto aquela ciência apresenta comprovada utilidade na modelagem de problemas envolvendo medições de natureza mais subjetiva. Em inspeção visual, por exemplo, o processo de classificação de ítens seriados segundo diferentes níveis de qualidade constitui uma forma de medição, porém, baseada nas assim chamadas escalas fracas — *escala nominal*, fundamento das operações de reconhecimento de padrões, e *escala ordinal*, determinante do processo de classificação, via seqüenciamento adequado dos padrões previamente identificados. Em contraste, nas atividades tradicionais de medição, geralmente se destaca apenas o aspecto exato (*sic*) do processo, cuja súmula são as medidas numéricas expressas na *escala racional*, apesar de nelas coexistirem, implicitamente, resultados parciais de medições realizadas no âmbito das outras escalas.

Em função do recente aumento da demanda por qualidade industrial, os laboratórios metrológicos têm sido obrigados a automatizar os seus ensaios, o que nem sempre pode ser realizado de forma simples, já que muitos procedimentos dependem de habilidades e sentidos eminentemente humanos, em geral difíceis de explicitar e/ou modelar. No caso específico do Laboratório de Metrologia do IPT, dentre a sua variada gama de ensaios, dois deles em particular — calibração de microescalas lineares e calibração de microtexturas de peneiras de medição — estão entre os mais demorados e desafiadores do ponto de vista da automação, pois, além de requererem instrumentação relativamente complexa, dependem fortemente da habilidade visual humana.

Para se automatizar esses ensaios utilizando-se técnicas de visão computacional, analisou-se inicialmente a fenomenologia do processo de captura de imagens, nas condições estabelecidas para os arranjos experimentais propostos e, paralelamente, se investigaram diversos paradigmas de medição, nas três escalas já referidas, aplicáveis à modelagem do processo de interpretação de cenas características de microescalas e de microtexturas de peneira. Dessa análise preliminar, resultaram diferentes modelos de medição, alguns dos quais emulando o processo evolutivo de aquisição de conhecimento visual humano, outros porém, de natureza puramente computacional.

Por causa da especificidade da aplicação, a tentativa de se construir os referidos modelos de medição a partir da composição de algoritmos de visão computacional disponíveis em bibliotecas comerciais mostrou-se infrutífera. Assim sendo, optou-se pela implementação de uma biblioteca própria, com todos os algoritmos necessários aos modelos citados, construíndo-se a partir destes 7 instrumentos metrológicos de visão computacional, dos quais 5 dedicados à calibração de microescalas e 2 à calibração de microtexturas de peneiras.

Considerando-se que o desempenho metrológico destes instrumentos, no tocante a exatidão e repetibilidade, seria afetado pela escolha dos parâmetros **P** inerentes aos algoritmos de visão computacional, e por certas decisões **Q** tomadas pelo operador por ocasião da captura de imagens, realizaram-se diversos experimentos para avaliação da real influência desse conjunto de parâmetros sobre a qualidade das medidas. Dessa forma, concluiu-se que, relativamente às decisões **Q** mencionadas acima: 1º) não são significativas as modificações ocasionadas por variações da potência de iluminação do sistema de captura de imagens; 2º) variações na distância focal, desde que inferiores a 0,06mm, podem ser ignoradas; 3º) a orientação do objeto no plano-imagem influi na exatidão e repetibilidade dos algoritmos, obtendo-se a melhor solução de compromisso,

para os casos da microescala e da microtextura de peneiras, respectivamente, quando as orientações dos traços da imagem são aproximadamente de 2 a 3° e 45°. Quanto aos parâmetros **P**, característicos dos instrumentos metrológicos de visão computacional e, em especial, das ferramentas de medição em escala nominal, os experimentos permitiram que se determinasse, para cada caso, um conjunto de valores que otimizava exatidão e repetibilidade, caracterizando-se, assim, os dois modelos de medição de microescalas e de microtexturas de peneiras. Além disso, esses experimentos tornaram evidente que o método de medição das escalas fracas exerce significativa influência sobre a qualidade final das medidas, fato que é completamente ignorado na avaliação das incertezas do processo convencional, a despeito de se reconhecer que diferentes operadores humanos não apresentam o mesmo grau de destreza e certamente não vêm a mesma cena da mesma maneira.

O modelo de medição de microescalas ficou, então, caracterizado, por medir as escalas *nominal*, *ordinal* e *racional*, respectivamente, através de: 1º) seleção dos pixels mediais entre as bordas claro→escuras e escuro→claras da imagem, limiarizadas e afinadas após aplicação do detector de Roberts; 2º) construção de seqüências ordenadas de conjuntos de pixels, utilizando a transformada de Hough aplicada às imagens de bordas; 3º) determinação das equações das retas através da aplicação do método dos mínimos quadrados às seqüências ordenadas de conjuntos de pixels da etapa anterior.

Para a microtextura de peneiras, dado que a sua calibração requer tanto a estimação dos valores médios da abertura da trama e diâmetro do fio quanto a inspeção de tolerâncias das aberturas, dois modelos de medição foram constituídos. No primeiro, as escalas *nominal*, *ordinal* e *racional* são determinadas via: 1º) limiarização automática da imagem, eliminação de objetos incompletos e localização das bordas através de um detector morfológico de bordas; 2º) construção de duas seqüências ordenadas de

conjuntos S_x e S_y agrupando os pixels das bordas segundo duas direções ortogonais; 3º) determinação das equações das retas, através da aplicação de mínimos quadrados aos pixels das seqüências de conjuntos S_x, S_y. No segundo modelo, voltado à inspeção de tolerâncias, a escala *nominal* é medida através de limiarização automática e as escalas *ordinal/racional*, através de operadores morfológicos utilizando elementos estruturantes do tipo *círculo*, com dimensões apropriadas.

Considerando-se que o fator de escala K, convertendo *nº de pixels* em mm, poderia ser obtido, para o caso de microescalas, seja através da imagem e das medidas correspondentes de uma microescala-referência previamente calibrada, seja através de um par de imagens da microescala-teste deslocada de uma distância D conhecida, realizaram-se experimentos para avaliação da repetibilidade do instrumento proposto, utilizando-se ambas essas técnicas. Os resultados desses experimentos revelaram que o novo instrumento apresenta repetibilidade (±0,1μm) duas vezes superior à do ensaio convencional (±0,2μm) quando a segunda técnica é adotada, mas esse desempenho pode ser 5 vezes superior (ou seja, ±0,04μm) caso o fator de escala seja calculado através da técnica baseada em imagem e medidas referência.

Com relação à calibração de peneiras, os experimentos realizados permitiram concluir que a repetibilidade do novo instrumento de estimação dos valores médios da abertura e do diâmetro do fio é da ordem de ±0,1μm, portanto 20 vezes maior que a do ensaio convencional (±2μm). Já o instrumento de inspeção de tolerâncias apresenta incerteza das medidas em torno de 2%; porém, não há como comparar esse resultado com alguma referência, já que, utilizando-se os métodos convencionais de medição, a inspeção de tolerâncias das aberturas de microtexturas de peneira pode ser considerada como uma atividade inviável do ponto de vista prático.

Analisando-se as eficiências dos novos instrumentos propostos, concluiu-se que, no caso da medição de microescalas, os dois métodos, baseados em medidas-referência e em deslocamento mecânico, são, respectivamente, 30 vezes e 15 vezes mais rápidos que o processo convencional. Já no caso da calibração de peneiras, o ganho é ainda mais significativo, pois os novos instrumentos realizam, em aproximadamente 1 hora, a medição e inspeção de tolerâncias da microtextura, utilizando uma amostra 13 vezes maior que a requerida pela norma ABNT-5734; dessa forma, pode-se definitivamente abandonar a norma ASTM E11-87, muito menos rigorosa, mas que era utilizada até então por ser extremamente exagerado o tempo requerido para se realizar o ensaio segundo a norma ABNT-5734.

Portanto, para sumarizar, pode-se afirmar que os resultados obtidos ao longo do desenvolvimento deste trabalho permitem concluir que:

- A visão computacional é uma ciência que pode ser aplicada às atividades de medição realizadas em laboratórios metrológicos de padrões secundários, com inúmeras vantagens em relação aos procedimentos metrológicos clássicos, pois: 1º) é capaz de realizar medições muito mais precisas e muito mais rápidas sem requerer, para tanto, instrumental mais sofisticado do que o que hoje já é utilizado; 2º) permite a medição de características de qualidade dos padrões que não podem, hoje, ser determinadas pelos métodos tradicionais.
- Por causa dos benefícios reportados anteriormente, a introdução dos instrumentos de medição baseados em visão computacional produzirá um impacto na comunidade dos laboratórios de padrões secundários, uma vez que as normas hoje adotadas terão que se completamente revistas, seja através da inclusão dos novos conceitos implícitos nos modelos de medição propostos, seja estabelecendo-se outros índices de avaliação da qualidade dos padrões, compatíveis com os recursos das novas ferramentas.

- A metodologia desenvolvida para construção dos instrumentos metrológicos de visão computacional, baseada no uso das escalas nominal, ordinal e racional, introduz um elo bastante forte entre os conceitos clássicos da visão computacional e a prática da metrologia, facilitando a modelagem de novos instrumentos computacionais para automação de diversos ensaios realizados em laboratórios metrológicos.

A despeito dos resultados obtidos, existem algumas recomendações que, se devidamente aplicadas, poderiam melhorar ainda mais o desempenho dos instrumentos metrológicos implementados.

Em particular, a identificação da *MTF* do sistema de aquisição de imagens, através de experimentos baseados na captura de imagens de bordas-degrau precisas, para diferentes orientações α, seria uma contribuição efetiva para se elaborar um filtro de reconstrução mais fiel. Aplicando-se, então, às imagens assim interpoladas, detectores regionais de bordas, conjuntamente com métodos alternativos de limiarização automática de imagens de bordas, seria de se esperar um real incremento na precisão das medidas, algo que não se obteve neste trabalho, em razão dos motivos expostos no capítulo 5.

Apesar de se utilizarem neste trabalho diversos paradigmas de medição, existem ainda muitos métodos de visão computacional que poderiam vir a ser proveitosamente explorados. O enquadramento de imagens é um exemplo típico: esse método poderia ser adotado como mais uma ferramenta de medição qualitativa, em que o desvio quadrático registrado entre a imagem-referência e a imagem-teste seria a medida da qualidade desta última.

Desejável também seria que se estudasse mais profundamente, para o caso da medição de microtexturas de peneiras, o processo de determinação do fator de escala K a partir de uma imagem-referência. Nesse caso, seria importante investigar o efeito das dimensões do objeto-referência sobre a precisão final das medidas.

Com relação a possíveis trabalhos futuros que venham a agregar conhecimento e eficiência às atividades dos laboratórios metrológicos de padrões secundários, através da incorporação de ferramentas de visão computacional, a lista é bastante extensa, podendo -se aqui citar os seguintes exemplos: calibração de escalas angulares (figura 6.1-a), automação de ensaios de dureza através de medição das imagens das impressões (figura 6.1-b), medição automática de rugosidade a partir de imagens geradas por microscópio de corte (figura 6.1-c).

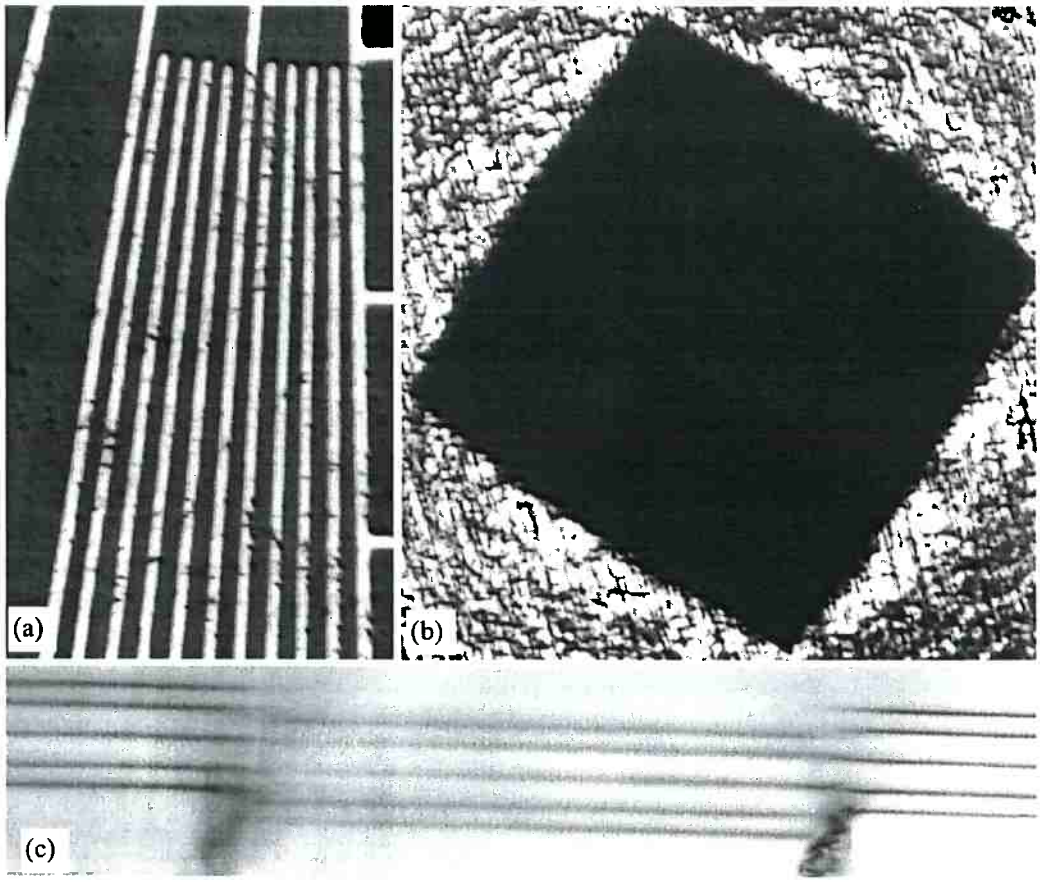

Figura 6.1: Cenas típicas de ensaios realizados em laboratórios metrológicos: (a) Escala angular; (b) Impressão de ensaio de dureza; (c) Franjas geradas em microscópio de corte, devidas a rugosidade.

REFERÊNCIAS BIBLIOGRÁFICAS

1. ABDOU, I.E.; PRATT, W.K. Quantitative design and evaluation of enhancement / thresholding edge detectors. **Proceedings of the IEEE**, v.67, no 5, p.753-763, May 1979.
2. ABDULLAH, F. *et al.* Modelling in measurement and instrumentation—an overview. **Measurement**, 14, p.41-54, 1994.
3. ABNT. Peneiras para ensaio com tela de tecido metálico. **Norma Brasileira** nº 5734. Associação Brasileira de Normas Técnicas, 1991.
4. ASTM. Wire-Cloth Sieves for Testing Purposes. **ASTM Standard E11-87**. ASTM, 1987.
5. BALLARD, D.H.; BROWN, C.M. **Computer Vision**. Englewood Cliffs, Prentice-Hall, 1982.
6. BECKWITH, T.G.; BUCK, N.L.; MARANGONI, R. **Mechanical Measurements**. Reading, Addison-Wesley, 1981.
7. BELAÏD, A. Metrology in quality control of nuts. In: **Proceedings of the International Conference on Pattern Recognition**. Atlantic City, 1990, p. 636-638.
8. BERKA, K. **Measurement: Its Concepts, Theories and Problems**. Dordrecht, D. Reidl Publishers, 1983.
9. BEYER, H.A. Linejitter and geometric calibration of CCD cameras. **ISPRS Journal of Photogrammetry and Remote Sensing**, 45, p.17-32, 1990.
10. BROWN, D.C. Close-range camera calibration. **Photogrammetric Engineering**, p. 855-866, 1971.
11. BUSCH, T. **Fundamentals of Dimensional Metrology**. 2ª ed. Delmar Publishers, 1987.
12. CAGLIOTI, V. On the uncertainty of straight lines in digital images. **CVGIP: Graphical Models and Image Processing**, v.55, nº 4, p.255-270, July 1993.
13. CALVERT, C. **Programando Aplicações em Windows com C e C++.** Rio de Janeiro, Berkeley Brasil Editora, 1994.
14. COURTNEY, P.; THACKER, N.; CLARK, A.F. Algorithmic modelling for performance evaluation. **Machine Vision and Applications**, 9, p.219-228, 1997.
15. DARWISH, A.M.; JAIN, A.K. A rule-based approach for visual pattern inspection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v.10, nº 1, p.56-68, Jan. 1988.
16. DOIRON, T.D. Computer vision based station for tool setting and tool form measurement. **Precision Engineering**, v.11, nº 4, p.231-238, 1989.
17. DOIRON, T.D. High precision gaging with computer vision systems. **Industrial Metrology**, 1, p.43-54, 1990.
18. DOM, B.E. *et al.* The P300: a system for automatic patterned wafer inspection. **Machine Vision and Applications**, 1, p.205-221, 1988.

19.EL-HAKIM, S.F. Some solutions to vision dimensional metrology problems. **Proceedings of the SPIE: Close-Range Photogrammetry Meets Machine Vision**, v.1395, p.480-487, 1990.

20.FERRIS, T.L.J. The concept of leap in measurement interpretation. **Measurement**, v.21, nº 4, p.137-146, 1997.

21.FLORY, R.E. Image acquisition technology. **Proceedings of the IEEE**, v.73, nº 4, p.613-637, Apr. 1985.

22.FILKENSTEIN, L. Measurement and instrumentation science—an analytical review. **Measurement**, 14, p.3-14, 1994.

23.FIOK, A.J.; BEK, J.; JAWORSKI, J.M. Some problems of measurement of real objects. In: **Proceedings of the 12th Triennial World Congress of the International Measurement Confederation**. Beijing, 1991, p.87-92.

24.GNIOTEK, K. Creative measurement in view of the scope of metrology. **Measurement**, v.20, nº4, p.259-266, 1997.

25.GONZALEZ, R.C.; WINTZ, P. **Digital Image Processing**. Reading, Addison-Wesley, 1987.

26.GHOSH, P.K. A solution of polygon containment, spatial planning and other related problems using Minkowski operations. **Computer Vision, Graphics and Image Processing**, 49, p.1-35, 1990.

27.GUO, R.; PANDIT, S.M. Automatic threshold selection based on histogram modes and a discrimination criterion. **Machine Vision and Applications**, 10, p.331-338, 1998.

28.HADDON, J.F. Generalized threshold selection for edge detection. **Pattern Recognition**, v.21, nº 3, p.195-203, 1988.

29.HARALICK, R.M. Machine vision mensuration. **Computer Vision, Graphics and Image Processing**, 40, p. 271-272, 1987.

30.HARALICK, R.M.; SHAPIRO, M. **Computer and Robot Vision**. Reading, Addison Wesley, 1991.

31.HOFMANN, D. Impact of measurement theory on scientific and technological progress. In: **Proceedings of the 12th Triennial World Congress of the International Measurement Confederation**. Beijing, 1991, p.20-29.

32.HOFMANN, D. Intelligent instruments: new solutions for old problems. **Measurement**, 13, p.23-37, 1994.

33.HOPP, T.H. Computational Metrology. **Manufacturing Review**, v.6, nº 4, p. 295-304, Dec. 1993.

34.HORN, B.K.P. **Robot Vision**. 2ª ed. Cambridge, MA, The MIT Press, 1986.

35.HUECKEL, M.H. A local visual operator which recognizes edges and lines. **Journal of the Association of Computing Machinery**, v.20, nº 4, p. 634-647, Oct. 1973.

36.HUNSICKER, R.J. *et al.* Automatic vision inspection and measurement system for external screw threads. **Journal of Manufacturing Systems**, v.13, nº 5, p.370-384, 1994.

37.KITCHEN, L.; ROSENFELD, A. Edge evaluation using local edge coherence. **IEEE Transactions on Systems, Man and Cybernetics**, v.SMC-11, nº 9, p.597-605, Sept. 1981.

38.KITTLER, J.; ILLINGWORTH, J. On threshold selection using clustering criteria. **IEEE Transactions on Systems, Man and Cybernetics**, v.SMC-15, nº 5, p.652-655, Sept/Oct 1985.

39.ILLINGWORTH, J.; KITLER, J. A survey of the Hough transform. **Computer Vision, Graphics and Image Processing**, 44, p.87-116, 1988.

40.IRVING, P. A. *et al.* Optical techniques for generation of component data suitable for input to CAD/CAM systems in the aerospace industry. In: **Advances in Manufacturing Technology**, v.IV, ed. Jeff Chandler, p.91-95. London, Kogan Page, 1989.

41.JENSEN, C.; ANASTASSIOU, D. Subpixel edge localization and the interpolation of still images. **IEEE Transactions on Image Processing**, v.4, nº 3, p.285-295, Mar 1995.

42.JIANG, B.C.; SHIAU, M.Y-R. A systematic methodology for determining / optimizing a machine vision system's capability. **Machine Vision and Applications**, 3, p.169-182, 1990.

43.JURAN, J.M; GRYNA, F.M. **Quality Planning and Analysis**. New York, McGraw-Hill Publishing Co. Ltd, 1978.

44.KEREN, D.;PELEG, S.; BRADA, R. Image sequence enhancement using subpixel displacements. **Proceedings of the IEEE**, p.742-746, 1988.

45.KHALAJ, B.H.; AGHAJAN, H.K.; KAILATH, T. Patterned wafer inspection by high resolution spectral estimation techniques. **Machine Vision and Applications**, 7, p.178-185, 1994.

46.KINGSLAKE, R. **Lens Design Fundamentals**. San Diego, Academic Press, 1978.

47.KLEIN, M.V.; FURTAK, T.E. **Optics**. 2ª ed. New York, John Wiley & Sons, 1986.

48.KROTKOV, E.P. Visual hyperacuity: representation and computation of high precision position information. **Computer Vision, Graphics and Image Processing**, 33, p.99-115, 1986.

49.LASSILA, A.; IKONEN, E.; RISKI, K. Interferometer for calibration of graduated line scales with a moving CCD camera as a line detector. **Applied Optics**, v.33, nº16, 1, p. 3600-3603, June 1994.

50.LENZ, R.K.; FRITSCH, D. Accuracy of videometry with CCD sensors. **ISPRS Journal of Photogrammetry and Remote Sensing**, 45, p.90-110, 1990.

51.LEVIALDI, S. Edge Extraction Techniques. In: **Fundamentals in Computer Vision**, ed. Faugeras, O.D., p.117-144. Cambridge, Cambridge University Press, 1983.

52.LOTUFO, R. A. Visão Computacional em Aplicações Industriais. **Apostila do 1º Seminário de Visão Computacional**. Instituto de Pesquisas Tecnológicas do Estado de São Paulo, 7 de maio de 1996.

53.LINK, W. **Metrologia Mecânica: Expressão da Incerteza de Medição**. São Paulo, Publicações Mitutoyo, 1997.

54.LYVERS, E.P. et al. Subpixel measurements using a moment-based edge operator. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v.11, nº 12, p.1293-1308, Dez. 1989.

55.MANDEVILLE, J.R. Novel method for analysis of printed circuit images. **IBM Journal of Research and Development**, v.29, nº 1, p.73-86, Jan. 1985.

56.MARI, L. The meaning of "quantity" in measurement. **Measurement**, v.17, nº 2, p. 127-138, 1996.

57.MARI, L. The role of determination and assignment in measurement. **Measurement**, v.21, nº 3, p.79-90, 1997.

58.MARION, A. **An Introduction to Image Processing**. 1st ed. London, Chapman and Hall, 1991.

59.MARR, D.; HILDRETH, E. Theory of edge detection. **Proceedings of the Royal Society of London**, v.B-207, p.187-217, 1980.

60.MARR, D. **Vision: A Computation Investigation into the Human Representation and Processing of Visual Information**. New York, W. H. Freeman, 1982.

61.MASCARENHAS, N.D.A. Introdução ao processamento digital de imagens. In: **Anais da 1ª Jornada EPUSP/IEEE em Computação Visual**. São Paulo, 4-7 Dez, 1990, p.387-420.

62.MICROSOFT. **Microsoft Visual C 2.0**. Reference and User's Guide. Redmond, WA, Microsoft Corporation, 1994.

63.MICROSOFT. **Microsoft Excel 7.0**. Reference and User's Guide. Redmond, WA, Microsoft Corporation, 1995.

64.MILLER, L. **Engineering Dimensional Metrology**. London, Edward Arnold Publisher, 1962.

65.MOORE, W.R. **Foundations of Mechanical Accuracy**. 1st ed. Bridgeport, The Moore Special Tool Company, 1970.

66.MUTECH. **M-Vision 1000.** Instalation and User's Guide. Woburn, MA, MuTech Corporation, 1995.

67.NALWA, V.S. Edge detector resolution improvement by image interpolation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v.PAMI-9, nº 3, p.446-451, May 1987.

68.NEVATIA, R.; BABU, R. Linear feature extraction and description. **Computer Graphics and Image Processing**, p.2567-269, 1980.

69.NIBLACK, W.; PETKOVIC, D. On improving the accuracy of the Hough Transform. **Machine Vision and Applications**, 3, p.87-106, 1990.

70.OKAWA, Y.; MIZUNO, S. Automatic inspection of diode pellets. **Machine Vision and Applications**, 4, p.131-133, 1991.

71.OLSEN, S.L. Estimation of noise in images: an evaluation. **CVGIP: Graphical Models and Image Processing**, v.55, nº 4, p.319-323, July 1993.

72.OTSU, N. A threshold selection method from gray-level histograms. **IEEE Transactions on Systems, Man and Cybernetics**, v.SMC-9, nº 1, p.62-66, Jan. 1979.

73.PALMER, P.L.; KITLER, J.; PETROU, M. An optimizing line finder using a Hough Transform algorithm. **Computer Vision and Image Understanding**, v.67, nº 1, p.1-23, Jul. 1997.

74.PETKOVIC, D.; NIBLACK, W.; FLICKNER, M. Projection-based high accuracy measurement of straight line edges. **Machine Vision and Applications**, 1, p.183-199, 1988.

75.PRATT, W.K. **Digital Image Processing**. New York, John Wiley & Sons, 1978.

76.PRESS, W.H. *et al.* **Numerical Recipes in C**. Cambridge, Cambridge University Press, 1988.

77.RAYNOR, J.M.; SEITZ, P. The technology and practical problems of pixel-synchronous CCD data acquisition for optical metrology applications. **Proceedings of the SPIE: Close-Range Photogrammetry Meets Machine Vision**, v.1395, p.96-103, 1990.

78.RISSE, T. Hough Transform for line recognition: complexity of evidence accumulation and cluster detection. **Computer Vision, Graphics and Image Processing**, 46, p.327-345, 1989.

79.RODRIGUEZ, A.A.; MANDEVILLE, J.R.; WU, Y. Calibration and alignment techniques for automated inspection of printed circuit patterns. **Industrial Metrology**, p.293-307, 1991.

80.RODRIGUEZ, A.A.; MANDEVILLE, J.R. Image registration for automated inspection of printed circuit patterns using CAD reference data. **Machine Vision and Applications**, 6, p.233-242, 1993.

81.ROSENFELD, A. Segmentation: Pixel-based methods. In: **Fundamentals in Computer Vision**, ed. Faugeras, O.D., p.225-237. Cambridge, Cambridge University Press, 1983.

82.ROSIN, P. L. Edges: saliency measures and automatic thresholding. **Machine Vision and Applications**, 9, p.139-159, 1997.

83.ROY, R.; KAILATH, T. ESPRIT — estimation of signal parameters via rotational invariance techniques. **IEEE Transactions on Acoustics, Speech and Signal Processing**, v.37, nº 7, p. 984-995, July 1989.

84.SAHOO, P.K *et al.* A survey of thresholding techniques. **Computer Vision, Graphics and Image Processing**, 41, p.233-260, 1988.

85.SANZ, J.L.C; PETKOVIC, D. Machine vision algorithms for automated inspection of thin-film disk heads. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v.10, nº 6, p.830-848, Nov. 1988.

86.SEITZ, P. Optical superresolution using solid-state cameras and digital signal processing. **Optical Engineering**, v.27, nº 7, p.535-540, July 1988.

87.SERRA, J. **Image Analysis and Mathematical Morphology**, v.1. 3rd ed. London, Academic Press, 1989.

88.STAHLBERG, L. Subpixel accuracy, gauging and flaw detection in machine vision. **Sensors**, p.32,33,35-37,88,90-91, May, 1995.

89.STAUNTON, R.C. Edge operator error estimation incorporating measurements of CCD TV camera transfer function. **IEE Proceedings of Visual Image Signal Processing**, v.145, nº 3, p.229-235, June 1998.

90.STRICKLAND, R.N.; DRAELOS, T.; MAO, Z. Edge detection in machine vision using a simple L_1 norm template matching algorithm. **Pattern Recognition**, v.23, nº 5, p.411-421, 1990.

91.SUBBARAO, M.; LU, M.C. Image sensing model and computer simulation for CCD camera systems. **Machine Vision and Applications**, 7, p.277-289, 1994.

92.SURESH, K.; VOELCKER, H.B. New challenges in dimensional metrology: a case study based on 'size'. **Manufacturing Review**, v.7, nº. 4, p.291-303, Dec. 1994.

93.THOMAS, G.G. **Engineering Metrology**. 1st ed. London, Butterworth, 1974.

94.TOMITA, Y.; HONDA, S. Experimental evaluation of instrument scale readings obtained by eye interpolation of fractional interval values. **Measurement**, 13, p.147-151, 1994.

95.TORRE, V.; POGGIO, T.A. On edge detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v.PAMI-8, nº 2, p.147-163, March 1986.

96.WESKA, J.S. A survey of threshold selection techniques. **Computer Graphics and Image Processing**, 7, p.259-265, 1978.

97.WOLBERG, G. **Digital Image Warping**. Washington DC, IEEE Computer Society Press, 1990.

98.YOUNG, R.A. Bridging the gap between vision and commercial applications. **Proceedings of the SPIE**, v.2411, p.2-14, 1995.

99.ZANDHUIS, J.A. *et al.* Sub-pixel non-parametric PSF estimation for image enhancement. **IEE Proceedings of Visual Image Signal Processing**, v.144, nº 5, Oct. 1997.

ANEXO I

Listagem dos Algoritmos Implementados

```
/*****************************************************************************************************
 *****************************************************************************************************
 INSTRUMENTOS METROLÓGICOS DE VISÃO COMPUTACIONAL

 Este programa implementa instrumentos metrológicos automáticos utilizando os algoritmos de visão
 computacional disponíveis no módulo Alg-VisaoComp.cpp.
 *****************************************************************************************************
 *****************************************************************************************************/

#pragma hdrstop
#include "Medicao.h"

/*****************************************************************************************************
 MedEscFreqEsp
 -------------
 Medição de microescala baseada no método de estimação de freqüências espaciais.
 Parâmetros:
   nImgIn : imagem original
   nImgOut: imagem de linhas equiespaçadas
   DirV   : TRUE-> medição ao longo da vertical
   distT  : distância nominal entre os traços da escala
 Variáveis globais:
   FeixeRetas
   ModMed
 *****************************************************************************************************/
void MedEscFreqEsp(int nImgIn, int nImgOut, BOOL DirV, double distT)
{
  double p;      // período fundamental
  int i;         // índice
  int n;         // número de retas
  double r0;     // distância da primeira reta à origem
  double ang;    // ângulo formado pelas retas com o eixo vertical
  double DH,DV; // projeções horizontal e vertical (em mm) da distância  de uma dada reta à origem

  // Estimação do período fundamental

  if (DirV)
    p = IdFreqESPRIT(nImgIn, 2, TRUE);
  else
    p = IdFreqESPRIT(nImgIn, 2, FALSE);

  // Cálculo das posições das retas da imagem e correspondentes medidas estimadas em mm

  r0  = FeixeRetas[nImgIn].Raio[0];
  n = FeixeRetas[nImgIn].nRetasImg;
  ModMed.bh = distT/p;
  for (i=0; i<n; i++) {
    ang = FeixeRetas[nImgIn].Angulo[i];
    FeixeRetas[nImgOut].Angulo[i] = ang;
    FeixeRetas[nImgOut].Raio[i] = r0+(double)i*p;
    DH  = ModMed.bv*(FeixeRetas[nImgOut].Raio[i]-r0)*cos(ang*GRAD);
    DV  = ModMed.bh*(FeixeRetas[nImgOut].Raio[i]-r0)*sin(ang*GRAD);
    FeixeRetas[nImgOut].MedCal[i] = sqrt(DH*DH+DV*DV);
    FeixeRetas[nImgOut].Medida[i] = (double)i*ModMed.bh;
  }
  FeixeRetas[nImgOut].nRetasImg = n;
  FeixeRetas[nImgIn].nRetasImg = 0;

  return;
}

/*****************************************************************************************************
  MedEscTHough
  ------------
  Medição da microescala utilizando a Transformada de Hough exclusivamente.
  Parâmetros:
    nImgIn : imagem da microescala
    nImgOut: imagem do feixe de retas localizado na microescala
    AngMin : ângulo mínimo de busca
    AngMax : ângulo máximo de busca
    DelAng : incremento angular
    Compri : comprimento do menor segmento de reta
    Esp    : espalhamento médio dos pixels em torno das retas
    dPicos : distância mínima entre picos no espaço de Hough
    FCorte : limiar de corte (% do maior pico)
    dHor   : TRUE-> medição ao longo da direção horizontal
  *****************************************************************************************************/
void MedEscTHough(int nImgIn,int nImgOut,double AngMin,double AngMax,double DelAng,double Compri,double Esp,
     int dPicos, double Fcorte, BOOL dHor)
{
  int nLin,nCol;     // número de linhas e de colunas da imagem
  int sImg;          // tamanho das imagens
  int i;             // índice
  HGLOBAL hM;        // manipulador de memória
```

```
  LPBYTE imgA;      // imagem apresentada

  // Alocação de memória

  nLin = DimBitmap[nImgIn].cy;
  nCol = DimBitmap[nImgIn].cx;
  sImg = nCol*nLin;
  hM = GlobalAlloc(GHND, (DWORD)sImg);
  imgA = (LPBYTE)GlobalLock(hM);

  // Inversão dos níveis de cinza da imagem

  for (i=0; i<sImg; i++) {
    PixImg[nImgIn][i] = abs(255-PixImg[nImgIn][i]);
    imgA[i] = ConvIndPal[PixImg[nImgIn][i]];
  }
  SetBitmapBits(hbm[nImgIn], (DWORD)sImg, (LPSTR)imgA);
  MostraImagem(nImgIn);

  // Localização das retas via Transformada de Hough

  DetCoordBordas(nImgIn);
  AplicaHough(nImgIn,nImgOut,FALSE,FALSE,TRUE,AngMin,AngMax,DelAng,Compri,Esp,TRUE,dPicos,FALSE,0,Fcorte);
  MostraImagem(nImgOut);
  AssociaImagemObjeto(nImgOut, TRUE, TRUE);

  // Recuperação da imagem original

  for (i=0; i<sImg; i++)
    PixImg[nImgIn][i] = abs(255-PixImg[nImgIn][i]);

  // Determinação do fator de conversão pixel->mm e das posições dos traços da microescala (em mm)

  ConvnPixelsmm(nImgOut, dHor);

  // Liberação de memória

  GlobalUnlock(hM);

  return;
}

/*****************************************************************************************************
 ConvnPixelsmm
 -------------
 Determinação do fator de conversão número de pixels -> mm e das posições das retas da imagem medidas em mm.
 Parâmetros:
   nImg: imagem do feixe de retas paralelas
   dHor: TRUE-> fator de escala ao longo da direção horizontal
 Variáveis globais:
   FeixeRetas
   ModMed
 *****************************************************************************************************/
void ConvnPixelsmm(int nImg, BOOL dHor)
{
  int i;        // índice
  int n;        // número de retas da imagem
  double FS;    // fator de conversão número de pixels->mm
  double rm;    // posição média do feixe de retas imagem
  double RM;    // posição média do feixe de retas objeto
  double err;   // erro associado à determinação do fator de conversão

  // Determinação do fator de conversão pixel->mm

  if (dHor) {
    ModMed.av = 0.0;
    ModMed.bv = 1.0/MapImgObj(nImg, &err);
    ModMed.cv = 0.0;
    FS = ModMed.bv;
  }
  else {
    ModMed.ah = 0.0;
    ModMed.bh = 1.0/MapImgObj(nImg, &err);
    ModMed.ch = 0.0;
    FS = ModMed.bh;
  }

  // Determinação das posições das retas da imagem nImg em mm

  n  = FeixeRetas[nImg].nRetasImg;
  rm = 0;
  for (i=0; i<n; i++) {
    rm+= FeixeRetas[nImg].Raio[i];
    RM+= FeixeRetas[nImg].Medida[i];
```

```
  }
  rm/= (double)n;
  RM/= (double)n;

  for (i=0; i<n; i++)
    FeixeRetas[nImg].MedCal[i] = RM+(FeixeRetas[nImg].Raio[i]-rm)*FS;
  InfoCaractImg(nImg,1,63);

  return;
}

/*****************************************************************************************************
 MapImgObj
 ---------
 Determina o fator de escala da imagem em uma dada direção utilizando um conjunto de distâncias no plano-
 imagem e suas medidas correspondentes no plano-objeto.
 Parâmetros:
   nImg: imagem de objeto descrito por conjunto de retas paralelas
   Ce  : erro associado à estimativa do fator de escala
 Retorna:
   fator de escala
 *****************************************************************************************************/
double MapImgObj(int nImg, double *Ce)
{
  int m;        // número de retas da imagem
  int i;        // índice
  double d;     // distância de uma reta à origem do sistema de coordenadas da imagem
  double D;     // distância correspondente a d, medida no plano-objeto
  double dm;    // distância de uma reta à posição média do feixe no plano-imagem
  double DM;    // distância de uma reta à posição média do feixe no plano-objeto
  double sm;    // variância da distância da "reta média" dda imagem
  int g;        // número de graus de liberdade usados no cálculo de sm
  double sigma; // variância da distância "d"
  double C;     // inverso da matriz (no caso, escalar) de erros da estimação
  double Q;     // produto [Bt]*[Cv]*[Theta] (no caso, escalar)
  double s;     // fator de escala estimado

  m = FeixeRetas[nImg].nRetasImg;

  C = 0.0;
  Q = 0.0;
  dm = 0;
  DM = 0;
  g = 0;
  for (i=0; i<m; i++) {
    dm+= FeixeRetas[nImg].Raio[i];
    DM+= FeixeRetas[nImg].Medida[i];
    sm+= FeixeRetas[nImg].Sigma[i]*FeixeRetas[nImg].Sigma[i]*(FeixeRetas[nImg].nPts[i]-1);
    g += FeixeRetas[nImg].nPts[i]-1;
  }
  dm/= (double)m;
  DM/= (double)m;
  sm/= (double)g;
  for (i=0; i<m; i++) {
    d = fabs(FeixeRetas[nImg].Raio[i]-dm);
    D = fabs(FeixeRetas[nImg].Medida[i]-DM);
    sigma = FeixeRetas[nImg].Sigma[i]*FeixeRetas[nImg].Sigma[i] + sm;
    C+= D*D/sigma;
    Q+= d*D/sigma;
  }
  s = fabs(Q/C);
  *Ce = 1.0/C;

  return s;
}

/*****************************************************************************************************
  MedEscMEvol1
  ------------
  Instrumento de medição da microescala baseado em processo evolutivo, utilizando:
  1) Ferramentas de medição em escala nominal: limiarização ou (limiarização+esqueletonização).
  2) Ferramentas de medição em escala ordinal: Transformada de Hough ou rotulação.
  3) Ferramentas de medição em escala racional: Transformada de Hough ou mínimos quadrados.
  Parâmetros:
    nImg1  : imagem da microescala
    nImg2  : imagem de trabalho
    nImg3  : imagem de trabalho
    limiar : valor do limiar de binarização
    iEsq   : TRUE-> aplicar esqueletonização
    ordTH  : TRUE-> usar Transformada de Hough nas medições em escala ordinal
    L      : comprimento do menor segmento de reta
    Esp    : espalhamento médio dos pixels em torno das retas
    dPicos : distância mínima entre picos no espaço de Hough
    AngMin : ângulo mínimo no espaço de Hough
```

```
    AngMax : ângulo máximo no espaço de Hough
    DelAng : incremento angular
    racMMQ : TRUE-> usar mínimos quadrados nas medições em escala racional
    iSigma1: TRUE-> adota Sigma(i) = 1
    dHor   : TRUE-> medição ao longo da direção horizontal
  ***************************************************************************************************/
void MedEscMEvol1(int nImg1,int nImg2,int nImg3,int limiar,BOOL iEsq,BOOL ordTH,double L,double Esp,
     double AngMin,double AngMax,double DelAng,int dPicos,double Fcorte,BOOL racMMQ,BOOL iSigma1,BOOL dHor)
{
  int i;            // índice
  int sImg;         // tamanho da imagem
  int nmasc;        // número de máscaras rotuladas
  HGLOBAL hM1;      // manipulador de memória
  LPBYTE imgOrig;   // imagem original

  // Alocação de memória

  sImg = DimBitmap[nImg1].cx*DimBitmap[nImg1].cy;
  hM1 = GlobalAlloc(GHND, (DWORD)sImg);
  imgOrig = (LPBYTE) GlobalLock(hM1);
  for (i=0; i<sImg; i++)
    imgOrig[i] = PixImg[nImg1][i];

  // Limiarização da imagem

  for (i=0; i<sImg; i++)
    if (PixImg[nImg1][i] > limiar)
      PixImg[nImg1][i] = 0;
    else
      PixImg[nImg1][i] = 255;
  sprintf(NomeImg[nImg1], "Imagem limiarizada");
  SetBitmapBits(hbm[nImg1], (DWORD)sImg, (LPSTR)PixImg[nImg1]);
  MostraImagem(nImg1);

  // Esqueletonização da imagem limiarizada quando requerido

  if (iEsq) {
    sprintf(NomeImg[nImg2], "Esqueleto da imagem");
    Esqueleto(nImg1, nImg2);
    for (i=0; i<sImg; i++)
      PixImg[nImg1][i] = PixImg[nImg2][i];
  }

  // Determinação da seqüência ordenada de grupos de pixels representativos de traços da microescala

  DetCoordBordas(nImg1);
  if (ordTH) {
    sprintf(NomeImg[nImg2], "Retas localizadas via T-H");
    AplicaHough(nImg1,nImg2,FALSE,TRUE,TRUE,AngMin,AngMax,DelAng,L,Esp,TRUE,dPicos,FALSE,0,Fcorte);
  }
  else {
    nmasc = MedOrdpRot(nImg1, nImg2, nImg3);
    sprintf(NomeImg[nImg2], " %d máscaras rotuladas", nmasc);
  }
  MostraImagem(nImg2);

  // Medição final em escala racional

  if (racMMQ) {
    if (ordTH)
      AjustaRetas1(nImg1, nImg2, nImg3, 1.5, FALSE);
    else {
      for (i=0; i<nmasc; i++)
        FeixeRetas[nImg2].Angulo[i] = (AngMin+AngMax)/2;
      AjustaRetas2(nImg1, nImg2, nImg3, -nmasc, FALSE);
    }
    MostraImagem(nImg3);
  }
  else {
    FeixeRetas[nImg3].nRetasImg = FeixeRetas[nImg2].nRetasImg;
    for (i=0; i<FeixeRetas[nImg3].nRetasImg; i++) {
      FeixeRetas[nImg3].Angulo[i] = FeixeRetas[nImg2].Angulo[i];
      FeixeRetas[nImg3].Raio  [i] = FeixeRetas[nImg2].Raio  [i];
      FeixeRetas[nImg3].nPts  [i] = FeixeRetas[nImg2].nPts  [i];
      FeixeRetas[nImg3].Sigma [i] = FeixeRetas[nImg2].Sigma [i];
    }
  }
  for (i=0; i<FeixeRetas[nImg3].nRetasImg; i++)
    if (iSigma1)
      FeixeRetas[nImg3].Sigma[i] = 1.0;
  AssociaImagemObjeto(nImg3, TRUE, TRUE);

  // Determinação do fator de conversão pixel->mm e das posições dos traços da microescala (em mm)
```

```
  ConvnPixelsmm(nImg3, dHor);

  // Recuperação da imagem original

  for (i=0; i<sImg; i++) {
    PixImg[nImg1][i] = imgOrig[i];
    imgOrig[i] = ConvIndPal[imgOrig[i]];
  }
  FeixeRetas[nImg1].nRetasImg = 0;
  SetBitmapBits(hbm[nImg1], (DWORD)sImg, (LPSTR)imgOrig);

  // Liberação de memória

  GlobalUnlock(hM1);

  return;
}

/*****************************************************************************************************
  MedEscMEvol2
  ------------
  Instrumento de medição da microescala baseado em processo evolutivo, utilizando:
  1) Ferramentas de medição em escala nominal: (reconstrução) + detecção de bordas
  2) Ferramentas de medição em escala ordinal: Transformada de Hough ou rotulação.
  3) Ferramentas de medição em escala racional: Transformada de Hough ou mínimos quadrados.
  Parâmetros:
    nImg1   : imagem da microescala
    nImg2   : imagem de trabalho
    nImg3   : imagem de trabalho
    W       : largura do filtro de suavização
    tDet    : detector de bordas: 0)Roberts; 1)Sobel; 2)Subpixel3; 3)Subpixel5
    iRecons: TRUE-> reconstruir imagem
    tBorda : 0: borda escuro-clara; 1: borda claro->escura; 2: eixo-medial
    L       : comprimento do menor segmento de reta
    Esp     : espalhamento médio dos pixels em torno das retas
    dPicos : distância mínima entre picos no espaço de Hough
    AngMin : ângulo mínimo no espaço de Hough
    AngMax : ângulo máximo no espaço de Hough
    DelAng : incremento angular
    Fcorte : limiar para o espaço de Hough
    racMMQ : TRUE-> usar mínimos quadrados nas medições em escala racional
    iSigma1: TRUE-> adota Sigma(i) = 1
    dHor    : TRUE-> medição ao longo da direção horizontal
    estN    : TRUE-> estima o ruído da imagem
 *****************************************************************************************************/
void MedEscMEvol2(int nImg1,int nImg2,int nImg3,int W,int tDet,BOOL iRecons,int tBorda,double L,double Esp,
     double AngMin,double AngMax,double DelAng,int dPicos,double Fcorte,BOOL racMMQ,BOOL iSigma1,BOOL dHor,
     BOOL estN)
{
  int sImg;          // tamanho das imagens
  int i;             // índice
  HGLOBAL hM1;       // manipulador de memória
  LPBYTE imgOrig;    // imagem original

  // Alocação de memória

  sImg = DimBitmap[nImg1].cx*DimBitmap[nImg1].cy;
  hM1 = GlobalAlloc(GHND, (DWORD)sImg);
  imgOrig = (LPBYTE) GlobalLock(hM1);
  for (i=0; i<sImg; i++)
    imgOrig[i] = PixImg[nImg1][i];
  ApagaImagem(nImg2);

  // Estimativa do ruído branco da imagem

  if (estN) {
    FiltroMedia(nImg1, nImg3, W);
    for (i=0; i<sImg; i++)
      PixImg[nImg3][i] = (BYTE)abs(PixImg[nImg1][i]-PixImg[nImg3][i]);
    SigmaN = EstatRuido(nImg3);
  }

  // Reconstrução da imagem, se requerido

  if (iRecons) {
    ApagaImagem(nImg3);
    Reconstrucao(nImg1, nImg3);
  }

  // Localização das retas sobre as bordas tipo tBorda da imagem

  ApagaImagem(nImg2);
  ApagaImagem(nImg3);
  if (tBorda!=2) {
```

```
    SeqAjusteRetas(nImg1,nImg2,nImg3,tDet,tBorda,0.01,AngMin,AngMax,DelAng,L,Esp,dPicos,Fcorte,racMMQ,
                   iRecons,FALSE);
    if (iSigma1)
      for (i=0; i<FeixeRetas[nImg3].nRetasImg; i++)
        FeixeRetas[nImg3].Sigma[i] = 1.0;
  }
  else {    // Localização dos eixos mediais

    // Localização das retas sobre as bordas escuro->claras da imagem

    SeqAjusteRetas(nImg1,nImg2,nImg3,tDet,0,0.01,AngMin,AngMax,DelAng,L,Esp,dPicos,Fcorte,racMMQ,
                   iRecons,FALSE);

    // Salvamento das medidas baseadas nas bordas escuro-claras da imagem

    FeixeRetas[nImg1].nRetasImg = FeixeRetas[nImg3].nRetasImg;
    for (i=0; i< FeixeRetas[nImg3].nRetasImg; i++) {
      FeixeRetas[nImg1].Raio  [i] = FeixeRetas[nImg3].Raio  [i];
      FeixeRetas[nImg1].Angulo[i] = FeixeRetas[nImg3].Angulo[i];
      FeixeRetas[nImg1].Medida[i] = FeixeRetas[nImg3].Medida[i];
      FeixeRetas[nImg1].nPts  [i] = FeixeRetas[nImg3].nPts  [i];
      if (iSigma1)
        FeixeRetas[nImg1].Sigma[i] = 1.0;
      else
        FeixeRetas[nImg1].Sigma[i] = FeixeRetas[nImg3].Sigma [i];
    }

    // Recuperação da imagem original e inicialização das imagens de trabalho nImg2 e nImg3

    for (i=0; i<sImg; i++)
      PixImg[nImg1][i] = imgOrig[i];
    ApagaImagem(nImg2);
    ApagaImagem(nImg3);

    // Localização das retas sobre as bordas claro->escuras da imagem

    SeqAjusteRetas(nImg1,nImg2,nImg3,tDet,1,0.01,AngMin,AngMax,DelAng,L,Esp,dPicos,Fcorte,TRUE,iRecons,TRUE);

    // Salvamento das medidas baseadas nas bordas claro-escuras da imagem

    FeixeRetas[nImg2].nRetasImg = FeixeRetas[nImg3].nRetasImg;
    for (i=0; i<FeixeRetas[nImg2].nRetasImg; i++) {
      FeixeRetas[nImg2].Raio  [i] = FeixeRetas[nImg3].Raio  [i];
      FeixeRetas[nImg2].Angulo[i] = FeixeRetas[nImg3].Angulo[i];
      FeixeRetas[nImg2].Medida[i] = FeixeRetas[nImg3].Medida[i];
      FeixeRetas[nImg2].nPts  [i] = FeixeRetas[nImg3].nPts  [i];
      if (iSigma1)
        FeixeRetas[nImg2].Sigma[i] = 1.0;
      else
        FeixeRetas[nImg2].Sigma[i] = FeixeRetas[nImg3].Sigma [i];
    }

    // Determinação das retas médias entre as bordas claro-escuras e escuro-claras da imagem

    FeixeMedio(nImg1, nImg2, nImg3);

  }

  InfoCaractImg(nImg3,1,42);

  // Determinação do fator de conversão pixel->mm e das posições das retas em mm

  ConvnPixelsmm(nImg3, dHor);

  // Recuperação da imagem original

  for (i=0; i<sImg; i++) {
    PixImg[nImg1][i] = imgOrig[i];
    imgOrig[i] = ConvIndPal[imgOrig[i]];
  }
  FeixeRetas[nImg1].nRetasImg = 0;
  SetBitmapBits(hbm[nImg1], (DWORD)sImg, (LPSTR)imgOrig);

  // Liberação de memória

  GlobalUnlock(hM1);

  return;
}

/*****************************************************************************************************************
  MedEscMEvol3
  ------------
  Instrumento de medição da microescala baseado em processo evolutivo, utilizando:
```

```
  1) Ferramenta inicial de medição em escala nominal: limiarização
  2) Ferramenta de medição em escala ordinal: Transformada de Hough ou rotulação.
  3) Ferramenta refinada de medição em escala nominal: detector especial de bordas
  3) Ferramentas de medição em escala racional: Transformada de Hough ou mínimos quadrados.
  Parâmetros:
    nImg1  : imagem da microescala
    nImg2  : imagem de trabalho
    nImg3  : imagem de trabalho
    dPtMin : TRUE-> detecção de níveis de cinza mínimos
    L      : comprimento do menor segmento de reta
    Esp    : espalhamento médio dos pixels em torno das retas
    dPicos : distância mínima entre picos no espaço de Hough
    angMin : ângulo mínimo no espaço de Hough
    angMax : ângulo máximo no espaço de Hough
    DelAng : incremento angular
    Fcorte : limiar para o espaço de Hough
    iSigma1: TRUE-> adota Sima(i) = 1
    dHor   : TRUE-> medição ao longo da direção horizontal
  ***************************************************************************************************/
void MedEscMEvol3(int nImg1, int nImg2, int nImg3, BOOL dPtMin, double L, double Esp, double angMin,
     double angMax, double DelAng, int dPicos, double Fcorte, BOOL iSigma1, BOOL dHor)
{
  int sImg;              // tamanho das imagens
  int i,k;               // índices
  HGLOBAL hM1;           // manipulador de memória
  LPBYTE imgOr;          // imagem de linhas horizontais original
  int lim;               // valor de limiar
  ponto ElemEst[25];     // elemento estruturante
  int cnz;               // nível de cinza da imagem
  float *Ydtr;           // máscara detectora dos traços da escala
  float xmin;            // abcissa correspondente ao mínimo de Ydtr

  // Alocação de memória para as imagens auxiliares

  sImg = DimBitmap[nImg1].cy*DimBitmap[nImg1].cx;
  hM1 = GlobalAlloc(GHND, (DWORD)sImg);
  imgOr = (LPBYTE) GlobalLock(hM1);

  // Salvamento da imagem original

  for (i=0; i<sImg; i++)
    imgOr[i] = PixImg[nImg1][i];

  // Limiariza a imagem original e preenche os vetores de coordenadas de bordas

  lim = CalcLimiar(nImg1);
  for (i=0; i<sImg; i++)
    if (PixImg[nImg1][i]<lim)
      PixImg[nImg1][i] = 255;
    else
      PixImg[nImg1][i] = 0;
  DetCoordBordas(nImg1);
  sprintf(NomeImg[nImg1], "Imagem limiarizada");
  SetBitmapBits(hbm[nImg1], (DWORD)sImg, (LPSTR)PixImg[nImg1]);
  MostraImagem(nImg1);

  // Obtém uma 1a. aproximação (nImg3) da posição dos eixos mediais dos traços via Transformada de Hough

  AplicaHough(nImg1,nImg3,FALSE,TRUE,TRUE,angMin,angMax,DelAng,L,Esp,TRUE,dPicos,FALSE,0,Fcorte);

  // Cria uma máscara binária (nImg2) a partir da imagem limiarizada (nImg1)

  k = CriaElemEstCirc(1, ElemEst);
  Dilatacao(nImg1, nImg2, k, ElemEst);
  sprintf(NomeImg[nImg2],"Máscara para seleção dos eixos mediais");
  SetBitmapBits(hbm[nImg2], (DWORD)sImg, (LPSTR)PixImg[nImg2]);
  MostraImagem(nImg2);

  // Acha a intersecção da imagem de retas nImg3 com a máscara nImg2

  for (i=0; i<sImg; i++)
    PixImg[nImg3][i] = __min(PixImg[nImg3][i],PixImg[nImg2][i]);
  sprintf(NomeImg[nImg3], "Segmentos mediais dos traços da escala");
  SetBitmapBits(hbm[nImg3], (DWORD)sImg, (LPSTR)PixImg[nImg3]);
  MostraImagem(nImg3);

  // Cria, a partir da máscara nImg2, a imagem rotulada nImg1 com os k objetos localizados em nImg2

  ApagaImagem(nImg1);
  sprintf(NomeImg[nImg1], "Máscara rotulada");
  k = Rotulacao(nImg2, nImg1, 0.1);
  ApagaImagem(nImg2);

  // Detecção da posição dos sub-pixels do eixo medial de cada traço rotulado
```

```
  Ydtr = vector(1,25);
  sprintf(NomeImg[nImg2], "Traços Detectados");
  for (cnz=255-k; cnz<255; cnz++) {
    if (!dPtMin) {
      xmin = GeraDetTraco(imgOr, nImg3, nImg1, cnz, dHor, Ydtr);
      DetTracoCorr(imgOr, nImg3, nImg1, nImg2, dHor, cnz, Ydtr, xmin);
    }
    else
      DetTracoPMin(imgOr, nImg3, nImg1, nImg2, dHor, cnz, Ydtr);
    SetBitmapBits(hbm[nImg2], (DWORD)sImg, (LPSTR)PixImg[nImg2]);
    MostraImagem(nImg2);
  }

  // Ajuste de retas através dos sub-pixels dos eixos mediais dos traços

  ApagaImagem(nImg1);
  AjustaRetas1(nImg2, nImg3, nImg1, 1.5, FALSE);

  // Cópia em nImg3 dos pixels correspondentes às retas ajustadas

  for (i=0; i<sImg; i++)
    PixImg[nImg3][i] = PixImg[nImg1][i];
  FeixeRetas[nImg3].nRetasImg = FeixeRetas[nImg1].nRetasImg;
  for (i=0; i<FeixeRetas[nImg3].nRetasImg; i++) {
    FeixeRetas[nImg3].Angulo[i] = FeixeRetas[nImg1].Angulo[i];
    FeixeRetas[nImg3].Raio  [i] = FeixeRetas[nImg1].Raio  [i];
    FeixeRetas[nImg3].nPts  [i] = FeixeRetas[nImg1].nPts  [i];
    if (iSigma1)
      FeixeRetas[nImg3].Sigma[i] = 1.0;
    else
      FeixeRetas[nImg3].Sigma[i] = FeixeRetas[nImg1].Sigma [i];
  }
  AssociaImagemObjeto(nImg3, TRUE, TRUE);

  // Determinação do fator de conversão pixel->mm e das posições dos traços da microescala (em mm)

  ConvnPixelsmm(nImg3, dHor);

  // Recuperação da imagem original

  for (i=0; i<sImg; i++) {
    PixImg[nImg1][i] = imgOr[i];
    imgOr[i] = ConvIndPal[imgOr[i]];
  }
  FeixeRetas[nImg1].nRetasImg = 0;
  SetBitmapBits(hbm[nImg1], (DWORD)sImg, (LPSTR)imgOr);

  // Liberação de memória

  GlobalUnlock(hM1);

  return;
}

/*****************************************************************************************************************
 SeqAjusteRetas
 --------------
 Sequência de operações para localizar feixe de retas em uma imagem.1) Detecção de bordas (claro-escuras ou
 escuro-claras) -> I1. 2) Transformada de Hough -> I2. 3) Ajuste de retas sobre os pontos de I1 usando a
 máscara I2.
 Ao final do processo, tem-se:
 nImg1 = imagem de bordas
 nImg2 = imagem de retas localizadas pela Transformada de Hough
 nImg3 = imagem de retas ajustadas
 FeixeRetas[nImg3] = posições das retas ajustadas
 Parâmetros:
   nImg1 : imagem original
   nImg2 : imagem de trabalho
   nImg3 : imagem de trabalho
   tDet  : 0-> Roberts;1-> Sobel;2-> subpixel 3x3;3-> subpixel 5x5
   tBorda: 0-> borda escuro-clara; 1-> borda claro-escura
   pcfBor: porcentagem tolerada de falsas bordas
   angMin: ângulo mínimo no espaço de Hough
   angMax: ângulo máximo no espaço de Hough
   dAng  : incremento angular no espaço de Hough
   L     : comprimento mínimo das retas da imagem
   Esp   : espessura das retas da imagem
   dPicos: distância mínima entre picos no espaço de Hough
   Fcorte: limiar de binarização do espaço de Hough
   AjRet : TRUE-> ajuste as retas
   Rec   : TRUE-> opera sobre a imagem reconstruída
   cEsReF: true-> calibração baseada em escala-referência
 Variáveis Globais:
```

```
   imgRec, FeixeRetas
 ***************************************************************************************************************/
void SeqAjusteRetas(int nImg1, int nImg2, int nImg3, int tDet, int tBorda, double pcfBor, double angMin,
     double angMax, double dAng, double L, double Esp, int dPicos, double Fcorte, BOOL AjRet, BOOL Rec,
     BOOL cEsRef)
{
  int i;    // índice
  int sImg; // tamanho da imagem

  // Detecção das bordas de nImg1

  switch (tDet) {
    case 0: DBordasRoberts(nImg1, nImg3, tBorda, TRUE, pcfBor, Rec); break;
    case 1: DBordasSobel(nImg1, nImg3, tBorda, TRUE, pcfBor, Rec); break;
    case 2: DBordasSubPixel3x3(nImg1, nImg3, tBorda, TRUE, pcfBor, Rec); break;
    case 3: DBordasSubPixel5x5(nImg1, nImg3, tBorda, TRUE, pcfBor, Rec); break;
  }
  sImg = DimBitmap[nImg1].cx*DimBitmap[nImg1].cy;
  for (i=0; i<sImg; i++)
    PixImg[nImg1][i] = PixImg[nImg3][i];

  // Localização das retas correspondentes via Transformada de Hough

  AplicaHough(nImg1,nImg2,Rec,TRUE,TRUE,angMin,angMax,dAng,L,Esp,TRUE,dPicos,FALSE,0,Fcorte);
  MostraImagem(nImg2);
  ApagaImagem(nImg3);

  // Ajuste das retas, quando requerido

  if (AjRet)
    AjustaRetas1(nImg1, nImg2, nImg3, 1.5, Rec);
  else {
    for (i=0; i<sImg; i++)
      PixImg[nImg3][i] = PixImg[nImg2][i];
    FeixeRetas[nImg3].nRetasImg = FeixeRetas[nImg2].nRetasImg;
    for (i=0; i<FeixeRetas[nImg3].nRetasImg; i++) {
      FeixeRetas[nImg3].Raio  [i] = FeixeRetas[nImg2].Raio  [i];
      FeixeRetas[nImg3].Angulo[i] = FeixeRetas[nImg2].Angulo[i];
      FeixeRetas[nImg3].Medida[i] = FeixeRetas[nImg2].Medida[i];
      FeixeRetas[nImg3].Sigma [i] = FeixeRetas[nImg2].Sigma [i];
      FeixeRetas[nImg3].nPts  [i] = FeixeRetas[nImg2].nPts  [i];
    }
  }

  // Associação das medidas da escala calibrada às retas localizadas na imagem

  if (cEsRef==0)    {
    AssociaImagemObjeto(nImg3, TRUE, TRUE);
    InfoCaractImg(nImg3,1,63);
  }

  return;
}

/***************************************************************************************************************
 MedEscpDesloc
 -------------
 Aplica uma sequência de operações de processamento de imagens a um conjunto de 2 imagens de uma mesma
 microescala, deslocada no plano-objeto de uma distância D conhecida. Após a determinação de bordas e a
 localização do eixo medial das retas de cada imagem, utiliza um método de estimação do fator de escala
 imagem->objeto.
 Parâmetros:
   nImg1   : imagem da escala-referência
   nImg2   : imagem da escala deslocada
   nImg3   : imagem de trabalho
   nImg4   : imagem de trabalho
   escRMed: escala-referência medida previamente
   distT   : distância nominal entre os traços da escala (em mm)
   D       : deslocamento no plano-objeto (em mm)
   fMedia  : TRUE-> filtro-média; FALSE-> filtro-mediana
   W       : largura do filtro de suavização
   tDet    : 0)Roberts; 1)Sobel; 2)Subpixel 3x3; 3)Subpixel 5x5
   L       : comprimento mínimo das retas da imagem
   Esp     : espessura das retas da imagem
   angMin  : limite inferior do intervalo de busca angular
   angMax  : limite superior do intervalo de busca angular
   DelAng  : incremento angular
   dPicos  : distância mínima entre picos no espaço de Hough
   Fcorte  : limiar de binarização do espaço de Hough
   dHor    : TRUE-> imagem deslocada aproximadamente na horizontal
 ***************************************************************************************************************/
void MedEscpDesloc(int nImg1, int nImg2, int nImg3, int nImg4, BOOL escRMed, double distT, BOOL fMedia,
     int W, int tDet, double L, double Esp, double angMin, double angMax, double DelAng, int dPicos,
     double Fcorte, double delta, BOOL dHor)
```

```
{
  int sImg;                  // tamanho das imagens
  int i,n;                   // índices
  HGLOBAL hM1,hM2;           // manipuladores de memória
  LPBYTE imgOr1,imgOr2;      // cópias das imagens originais nImg1 e nImg2
  double FS;                 // fator de escala
  double er;                 // não utilizado
  double Rcg;                // posição radial (em pixels) do centro de gravidade do feixe de retas

  // Salvamento das imagens originais: imgOr1, imgOr2

  sImg = DimBitmap[nImg1].cy*DimBitmap[nImg1].cx;
  hM1 = GlobalAlloc(GHND, (DWORD)sImg);
  imgOr1 = (LPBYTE) GlobalLock(hM1);
  hM2 = GlobalAlloc(GHND, (DWORD)sImg);
  imgOr2 = (LPBYTE) GlobalLock(hM2);
  for (i=0; i<sImg; i++) {
    imgOr1[i] = PixImg[nImg1][i];
    imgOr2[i] = PixImg[nImg2][i];
  }

  // Localização dos eixos mediais das retas das imagens nImg1 e nImg2

  if (escRMed) {
    // se a imagem nImg1 da microescala já foi medida apenas estima o ruído
    if (fMedia)
      FiltroMedia(nImg1, nImg2, W);
    else
      FiltroMediana(nImg1, nImg2, W);
    for (i=0; i<sImg; i++)
      PixImg[nImg2][i] = (BYTE)abs(PixImg[nImg1][i]-PixImg[nImg2][i]);
    SigmaN = EstatRuido(nImg2);
  }
  else {
    // Mede as posições das retas da microescala (nImg1)
    for (i=0; i<sImg; i++)
      imgRec[i] = PixImg[nImg1][i];
    MedEscMEvol2(nImg1,nImg2,nImg3,W,tDet,FALSE,2,L,Esp,angMin,angMax,DelAng,dPicos,Fcorte,TRUE,FALSE,dHor,
                 TRUE);
  }
  // Mede as posições das retas da microescala deslocada (nImg2)
  for (i=0; i<sImg; i++)
    PixImg[nImg1][i] = imgOr2[i];
  ApagaImagem(nImg2);
  ApagaImagem(nImg4);
  for (i=0; i<sImg; i++)
    imgRec[i] = PixImg[nImg1][i];
  MedEscMEvol2(nImg1,nImg2,nImg4,W,tDet,FALSE,2,L,Esp,angMin,angMax,DelAng,dPicos,Fcorte,TRUE,FALSE,dHor,
               FALSE);

  // Determinação do fator de escala

  if (dHor) {
    ModMed.bv = 1.0/EstimaFEsc(nImg3, nImg4, delta, &er);
    FS = ModMed.bv;
  }
  else {
    ModMed.bh = 1.0/EstimaFEsc(nImg3, nImg4, delta, &er);
    FS = ModMed.bh;
  }

  // Medição da microescala

     // Reta-Referência: nImg3

  Rcg = 0.0;
  n = FeixeRetas[nImg3].nRetasImg;
  for (i=0; i<n; i++)
    Rcg+= FeixeRetas[nImg3].Raio[i];
  Rcg = Rcg/(double)n;
  for (i=0; i<n; i++)   {
    FeixeRetas[nImg3].Medida[i] = i*distT;
    FeixeRetas[nImg3].MedCal[i] = (FeixeRetas[nImg3].Raio[i]-Rcg)*FS;
  }
  InfoCaractImg(nImg3,1,63);

    // Reta-Deslocada: nImg4

  Rcg = 0.0;
  n = FeixeRetas[nImg4].nRetasImg;
  for (i=0; i<n; i++)
    Rcg+= FeixeRetas[nImg4].Raio[i];
  Rcg = Rcg/(double)n;
  for (i=0; i<n; i++)   {
```

```
    FeixeRetas[nImg4].Medida[i] = i*distT;
    FeixeRetas[nImg4].MedCal[i] = (FeixeRetas[nImg4].Raio[i]-Rcg)*FS;
  }
  InfoCaractImg(nImg4,1,63);

  // Recuperação das imagens originais

  for (i=0; i<sImg; i++) {
    PixImg[nImg1][i] = imgOr1[i];
    imgOr1[i] = ConvIndPal[imgOr1[i]];
    PixImg[nImg2][i] = imgOr2[i];
    imgOr2[i] = ConvIndPal[imgOr2[i]];
  }
  SetBitmapBits(hbm[nImg1], (DWORD)sImg, (LPSTR)imgOr1);
  SetBitmapBits(hbm[nImg2], (DWORD)sImg, (LPSTR)imgOr2);
  sprintf(NomeImg[nImg1], "Imagem %d", nImg1);
  sprintf(NomeImg[nImg2], "Imagem %d", nImg2);
  FeixeRetas[nImg1].nRetasImg = 0;
  FeixeRetas[nImg2].nRetasImg = 0;

  // Liberação de memória

  GlobalUnlock(hM1);
  GlobalUnlock(hM2);

  return;
}

/*****************************************************************************************************
 GeraDetTraco
 ------------
 Gera uma máscara de convolução para a detecção de um  traço da imagem da microescala. A máscara gerada
 baseia-se nos valores médios dos níveis de cinza observados na região ao redor do traço.
 Parâmetros:
   imgEsc: pixels da imagem da escala
   nImgTr: número da imagem segmentada contendo os traços da escala
   nImgRt: número da imagem contendo regiões ao redor dos traços
   ntraco: número do traço
   dHor  : TRUE-> escala com os traços na vertical
   Ydtr  : vetor com os valores de detector de traços
 Retorna:
   xmin  : posição x tal que Ydtr é mínimo
 *****************************************************************************************************/
float GeraDetTraco(LPBYTE imgEsc, int nImgTr, int nImgRt, int ntraco, BOOL dHor, float *Ydtr)
{
  int nLin, nCol;       // número de linhas e de colunas da imagem
  int i,j,k,m,n;        // índices
  int a,b,c,d,e,f,g;    // índices
  int lin1,col1;        // deslocamentos de uma linha e de uma coluna
  int lin2,col2;        // deslocamentos de duas linhas e de duas colunas
  int lin3,col3;        // deslocamentos de 3 linhas e de 3 colunas
  int lin4,col4;        // deslocamentos de 4 linhas e de 4 colunas (não utilizados)
  float fx,fy;          // fatores para identificação da direção (x ou y) relevante (não utilizados)
  float *X, *Y;         // perfil representativo da seção transversal dos traços
  float *Y2;            // vetor de derivadas segundas da função acima
  float xp,yp;          // valores interpolados desse perfil
  float xmin,ymin;      // ponto de mínimo do perfil

  // Aloca memória para as coordenadas da máscara (1x7) inicial

  nLin = DimBitmap[nImgTr].cy;
  nCol = DimBitmap[nImgTr].cx;
  X  = vector(1,7);
  Y  = vector(1,7);
  Y2 = vector(1,7);
  for (i=1; i<=7; i++) {
    X[i] = (float)(i-4);
    Y[i] = 0.0F;
  }

  // Cria os fatores básicos para a seleção dos pixels da imagem da escala

  GeraIndPxViz(dHor,&lin1,&lin2,&lin3,&lin4,&col1,&col2,&col3,&col4,&fx,&fy);

  // Obtém os níveis médios de cinza ao longo de 7 pixels aproximadamente
  // perpendiculares à direção do traço de número "ntraco" da imagem

  n = 0;
  for (i=3; i<nLin-3; i++)
    for (j=3; j<nCol-3; j++) {
      d = i*nCol+j;
      if ((PixImg[nImgTr][d]==255) && (PixImg[nImgRt][d]==ntraco)) {
        a = (i-lin3)*nCol+(j-col3);
        b = (i-lin2)*nCol+(j-col2);
```

```
        c = (i-lin1)*nCol+(j-col1);
        e = (i+lin1)*nCol+(j+col1);
        f = (i+lin2)*nCol+(j+col2);
        g = (i+lin3)*nCol+(j+col3);
        n++;
        Y[1]+= (float)imgEsc[a];
        Y[2]+= (float)imgEsc[b];
        Y[3]+= (float)imgEsc[c];
        Y[4]+= (float)imgEsc[d];
        Y[5]+= (float)imgEsc[e];
        Y[6]+= (float)imgEsc[f];
        Y[7]+= (float)imgEsc[g];
      }
    }

  // Determina os 7 valores da máscara e identifica o menor desses valores

  for (k=1; k<=7; k++)
    Y[k]/= (float)(255*n);
  ymin = 255.0F;
  m = 1;
  for (k=1; k<=7; k++) {
    if (Y[k]<ymin) {
      ymin = Y[k];
      xmin = (float)(k-4);
    }
    Ydtr[m] = Y[k];
    m+= 4;
  }

  // Obtém valores interpolados para a máscara

  spline(X,Y,7,Y[2]-Y[1],Y[7]-Y[6],Y2);
  xp = -2.75F;
  m = 2;
  while (xp<3.0F) {
    for (k=1; k<=3; k++) {
      splint(X,Y,Y2,7,xp,&yp);
      if (yp<ymin) {
        ymin = yp;
        xmin = xp;
      }
      Ydtr[m] = yp;
      m++;
      xp+= 0.25F;
    }
    xp+= 0.25F;
    m++;
  }

  return(xmin);
}

/*****************************************************************************************************
 GeraIndPxViz
 ------------
 Gera, de acordo com a orientação dada (vertical ou horizontal), os fatores utilizados no cálculo dos
 índices (linha e coluna) de um dado pixel.
 Parâmetros:
   dHor     : TRUE-> reta vertical
   lin1,col1: deslocamentos de uma linha e de uma coluna
   lin2,col2: deslocamentos de duas linhas e de duas colunas
   lin3,col3: deslocamentos de 3 linhas e de 3 colunas
   lin4,col4: deslocamentos de 4 linhas e de 4 colunas
   fx,fy    : fatores para identificação da direção (x ou y) relevante
 *****************************************************************************************************/
void GeraIndPxViz(BOOL dHor,int *lin1,int *lin2,int *lin3,int *lin4,int *col1,int *col2,int *col3,int *col4,
                  float *fx,float *fy)
{
  if (dHor) {   // Traços verticais: seleciona pixels na horizontal
    *lin4 = 0;
    *col4 = 4;
    *lin3 = 0;
    *col3 = 3;
    *lin2 = 0;
    *col2 = 2;
    *lin1 = 0;
    *col1 = 1;
    *fx   = 1.0F;
    *fy   = 0.0F;
  }
  else {        // Traços horizontais: seleciona pixels na vertical
    *lin4 = 4;
    *col4 = 0;
```

```
    *lin3 = 3;
    *col3 = 0;
    *lin2 = 2;
    *col2 = 0;
    *lin1 = 1;
    *col1 = 0;
    *fx   = 0.0F;
    *fy   = 1.0F;
  }

  return;
}

/*****************************************************************************************************
 DetTracoCorr
 ------------
 Efetua a convolução da imagem da microescala com o detector unidimensional de traços, localizando, com
 precisão de subpixel, as coordenadas do centro do traço.
 Parâmetros:
   imgEsc: imagem da microescala
   nImgTr: número da imagem dos eixos mediais aproximados da escala
   nImgRt: número da imagem contendo regiões em torno dos eixos
   nImgF :  número da imagem dos eixos mediais "exatos"
   dHor  : TRUE-> escala com os traços na vertical
   ntraco: número do traço considerado
   Ydtr  : detector de traços
   xmin  : abcissa correspondente ao menor valor de Ydtr
 *****************************************************************************************************/
void DetTracoCorr(LPBYTE imgEsc, int nImgTr, int nImgRt, int nImgF, BOOL dHor, int ntraco,
                  float *Ydtr, float xmin)
{
  int nLin, nCol;        // número de linhas e de colunas da imagem
  int i,j,k;             // índices
  int a,b,c,d,e,f,g;     // índices
  int lin1,col1;         // deslocamentos de uma linha e de uma coluna
  int lin2,col2;         // deslocamentos de duas linhas e de duas colunas
  int lin3,col3;         // deslocamentos de três linhas e de três colunas
  int lin4,col4;         // deslocamentos de quatro linhas e de quatro colunas
  float fx,fy;           // fatores para identificação da direção (x ou y) relevante
  float pmax;            // valor máximo da função convolução
  float p[15];           // valores da função convolução p = p(x,y)
  float x[15],y[15];     // coordenadas x,y da função convolução

  nLin = DimBitmap[nImgTr].cy;
  nCol = DimBitmap[nImgTr].cx;

  // Cria os índices básicos para a seleção dos pixels da imagem da escala

  GeraIndPxViz(dHor,&lin1,&lin2,&lin3,&lin4,&col1,&col2,&col3,&col4,&fx,&fy);

  // Efetua a convolução da máscara detectora com a imagem de um dado traço da escala
  // Notar que a cada traço da escala corresponde uma determinada máscara detectora

  for (i=3; i<nLin-3; i++)
    for (j=3; j<nCol-3; j++) {
      d = i*nCol+j;
      if ((PixImg[nImgTr][d]==255) && (PixImg[nImgRt][d]==ntraco)) {
        // Identifica os pixels vizinhos ao pixel central da máscara
        a = (i-lin3)*nCol+(j-col3);
        b = (i-lin2)*nCol+(j-col2);
        c = (i-lin1)*nCol+(j-col1);
        e = (i+lin1)*nCol+(j+col1);
        f = (i+lin2)*nCol+(j+col2);
        g = (i+lin3)*nCol+(j+col3);
        // Efetua a convolução da imagem com a máscara detectora ao longo de
        // uma varredura abrangendo o pixel central e os dois pixels vizinhos
        ConvUnidEsp(imgEsc,Ydtr,xmin,fx,fy,a,b,c,d,e,f,g,0,p,x,y);
        ConvUnidEsp(imgEsc,Ydtr,xmin,fx,fy,(i-lin4)*nCol+(j-col4),a,b,c,d,e,f,5,p,x,y);
        ConvUnidEsp(imgEsc,Ydtr,xmin,fx,fy,a,b,c,d,e,f,(i+lin4)*nCol+(j+col4),10,p,x,y);
        // Identifica o eixo medial do traço com a posição da máscara
        // correspondente ao ponto de máximo da função convolução
        pmax = -1.0F;
        for (k=0; k<15; k++)
          if (p[k]>pmax) {
            pmax = p[k];
            CoordBordas.XBorda[d] = (float)j+x[k];
            CoordBordas.YBorda[d] = (float)i+y[k];
          }
        // Acrescenta à imagem final o pixel correspondente a um ponto do eixo medial do traço
        k = (int)CoordBordas.YBorda[d]*nCol+(int)CoordBordas.XBorda[d];
        PixImg[nImgF][k] = 255;
      }
    }
```

```
  return;
}

/*****************************************************************************************************
 DetTracoPMin
 ------------
 Localiza, com precisão de sub-pixel, os pontos do eixo medial de cada traço. Admite que esses pontos
 correspondem ao mínimo da curva ajustada ao longo de 7 pixels tomados na direção normal à direção do
 eixo medial aproximado,obtido previamente via Transformada de Hough.
 Parâmetros:
   imgEsc: imagem da microescala
   nImgTr: número da imagem dos eixos mediais aproximados da microescala
   nImgRt: número da imagem contendo regiões em torno dos eixos
   nImgF :  número da imagem dos eixos mediais "exatos"
   dHor  : TRUE-> escala com os traços na vertical
   ntraco: número do traço considerado
   Ydtr  : detector de traços
 *****************************************************************************************************/
void DetTracoPMin(LPBYTE imgEsc, int nImgTr, int nImgRt, int nImgF, BOOL dHor, int ntraco, float *Ydtr)
{
  int nLin, nCol;        // número de linhas e de colunas da imagem
  int i,j,k,m;           // índices
  int a,b,c,d,e,f,g;     // índices
  int lin1,col1;         // deslocamentos de uma linha e de uma coluna
  int lin2,col2;         // deslocamentos de duas linhas e de duas colunas
  int lin3,col3;         // deslocamentos de três linhas e de três colunas
  int lin4,col4;         // deslocamentos de quatro linhas e de quatro colunas (não utilizados)
  float *X, *Y;          // perfil de níveis de cinza de uma seção transversal de um dado traço
  float *Y2;             // vetor de derivadas segundas da função acima
  float xp,yp;           // pontos interpolados ao longo dessa função
  float xmin,ymin;       // ponto de mínimo da referida função
  float fx,fy;           // fatores para identificação da direção (x ou y) relevante
  int ix,iy;             // valores inteiros das coordenadas do eixo medial dos traços

  // Alocação de memória para os vetores representativos da função Nível de Cinza em uma dada direção

  nLin = DimBitmap[nImgTr].cy;
  nCol = DimBitmap[nImgTr].cx;
  X  = vector(1,7);
  Y  = vector(1,7);
  Y2 = vector(1,7);
  for (i=1; i<=7; i++) {
    X[i] = (float)(i-4);
    Y[i] = 0.0F;
  }

  // Cria os índices básicos para a seleção dos pixels da imagem da escala

  GeraIndPxViz(dHor,&lin1,&lin2,&lin3,&lin4,&col1,&col2,&col3,&col4,&fx,&fy);

  // Efetua a convolução da máscara detectora com a imagem de um dado traço da escala
  // Notar que a cada traço da escala corresponde uma determinada máscara detectora

  for (i=3; i<nLin-3; i++)
    for (j=3; j<nCol-3; j++) {
      d = i*nCol+j;
      if ((PixImg[nImgTr][d]==255) && (PixImg[nImgRt][d]==ntraco)) {
        // Identifica os pixels vizinhos ao pixel central da máscara
        a = (i-lin3)*nCol+(j-col3);
        b = (i-lin2)*nCol+(j-col2);
        c = (i-lin1)*nCol+(j-col1);
        e = (i+lin1)*nCol+(j+col1);
        f = (i+lin2)*nCol+(j+col2);
        g = (i+lin3)*nCol+(j+col3);
        // Constrói um perfil de níveis de cinza na direção normal à reta aproximada
        Y[1]+= (float)imgEsc[a];
        Y[2]+= (float)imgEsc[b];
        Y[3]+= (float)imgEsc[c];
        Y[4]+= (float)imgEsc[d];
        Y[5]+= (float)imgEsc[e];
        Y[6]+= (float)imgEsc[f];
        Y[7]+= (float)imgEsc[g];
        // Ajusta uma curva spline ao longo do perfil de níveis de cinza
        // Localiza o ponto de mínimo dessa curva
        spline(X,Y,7,Y[2]-Y[1],Y[7]-Y[6],Y2);
        xp = -2.75F;
        m = 2;
        ymin = 255.0F;
        while (xp<3.0F) {
          for (k=1; k<=3; k++) {
            splint(X,Y,Y2,7,xp,&yp);
            if (yp<ymin) {
              ymin = yp;
              xmin = xp;
```

```
            }
            Ydtr[m] = yp;
            m++;
            xp+= 0.25F;
          }
          xp+= 0.25F;
          m++;
        }
        CoordBordas.XBorda[d] = (float)j+xmin*fx;
        CoordBordas.YBorda[d] = (float)i+xmin*fy;
        // Acrescenta à imagem final o pixel correspondente a um ponto do eixo medial do traço
        ix = (CoordBordas.XBorda[d]-floor(CoordBordas.XBorda[d])<0.5) ?
             (int)CoordBordas.XBorda[d] : (int)CoordBordas.XBorda[d]+1;
        iy = (CoordBordas.YBorda[d]-floor(CoordBordas.YBorda[d])<0.5) ?
             (int)CoordBordas.YBorda[d] : (int)CoordBordas.YBorda[d]+1;
        PixImg[nImgF][iy*nCol+ix] = 255;
      }
    }

  return;
}

/*****************************************************************************************************
 ConvUnidEsp
 -----------
 Aplica uma convolução local a uma região da imagem limitada a apenas 7 pixels vizinhos escolhidos ao longo
 de uma das  direção x ou y. A máscara adotada contém 25 valores, dos quais 7 correspondem a pixel e os
 demais valores a 1/4 de pixel. Adota-se uma varredura de 1/4 em 1/4 pixel, de modo que a convolução
 abrange 5 pontos: o pixels central e seus dois subpixels vizinhos (+1/4,+1/2,-1/4,-1/2).
 Parâmetros:
   img  : imagem
   M    : máscara de convolução
   xmin : posição do valor mínimo da máscara M relativamente ao pixel central
   fx,fy: fatores seletores da direção (x ou y) relevante
   d    : índice do pixel central
   a,b,c: índices dos pixels vizinhos anteriores (acima ou à esquerda)
   e,f,g: índices dos pixels vizinhos posteriores (abaixo ou à direita)
   n    : índice inicial dos vetores p,x,y
   p    : vetor com os valores da função convolução p = p(x,y)
   x,y  : coordenadas da função p
 *****************************************************************************************************/
void ConvUnidEsp(LPBYTE img,float *M,float xmin,float fx,float fy,int a,int b,int c,int d,int e,int f,int g,
                 int n,float *p,float *x,float *y)
{
  int i1,i2,i3,i4,i5;       // índices

  i1 = n;
  i2 = n+1;
  i3 = n+2;
  i4 = n+3;
  i5 = n+4;

  // Máscara sem deslocamento
  p[i1] = (float)(img[a]*M[1]+img[b]*M[5]+img[c]*M[9]+img[d]*M[13]+img[e]*M[17]+img[f]*M[21]+img[g]*M[25]);
  x[i1] = xmin*fx;
  y[i1] = xmin*fy;

  // Máscara deslocada de 1/4 de pixel abaixo ou à direita
  p[i2] = (float)(img[a]*M[1]+img[b]*M[4]+img[c]*M[8]+img[d]*M[12]+img[e]*M[16]+img[f]*M[20]+img[g]*M[24]);
  x[i2] = (0.25F+xmin)*fx;
  y[i2] = (0.25F+xmin)*fy;

  // Máscara deslocada de 1/5 pixel abaixo ou à direita
  p[i3] = (float)(img[a]*M[1]+img[b]*M[3]+img[c]*M[7]+img[d]*M[11]+img[e]*M[15]+img[f]*M[19]+img[g]*M[23]);
  x[i3] = (0.50F+xmin)*fx;
  y[i3] = (0.50F+xmin)*fy;

  // Máscara deslocada de 1/4 de pixel acima ou à esquerda
  p[i4] = (float)(img[a]*M[2]+img[b]*M[6]+img[c]*M[10]+img[d]*M[14]+img[e]*M[18]+img[f]*M[22]+img[g]*M[25]);
  x[i4] = (-0.25F+xmin)*fx;
  y[i4] = (-0.25F+xmin)*fy;

  // Máscara deslocada de 1/5 pixel acima ou à esquerda
  p[i5] = (float)(img[a]*M[3]+img[b]*M[7]+img[c]*M[11]+img[d]*M[15]+img[e]*M[19]+img[f]*M[23]+img[g]*M[25]);
  x[i5] = (-0.50F+xmin)*fx;
  y[i5] = (-0.50F+xmin)*fy;

  return;
}

/*****************************************************************************************************
 EstimaFEsc
 ----------
 Determina o fator de escala da imagem em uma das direções x ou y, utilizando um conjunto de observações
```

```
 correspondentes à mesma distância D (em mm). A escolha da direção depende apenas da seleção adequada das
 imagens.
 Parâmetros:
   nImg1: imagem referência
   nImg2: imagem deslocada
   D    : deslocamento conhecido em mm em uma das direções x ou y
   Ce   : erro associado à estimativa do fator de escala
 Retorna:
   fator de escala
 ***************************************************************************************************************/
double EstimaFEsc(int nImg1, int nImg2, double D, double *Ce)
{
  int m;        // número de distâncias entre retas correspondentes nas imagens 1 e 2
  int i;        // índice
  double d;     // distância entre retas correspondentes nas duas imagens
  double sigma; // variância da distância "d"
  double C;     // inverso da matriz (no caso, escalar) de erros da estimação
  double Q;     // produto [Bt]*[Cv]*[Theta] (no caso, escalar)
  double s;     // fator estimado de amplificação da imagem
  double Dp;    // deslocamento em mm na direção das retas da imagem

  m = FeixeRetas[nImg2].nRetasImg;
  Dp = fabs(D/sin(FeixeRetas[nImg1].Angulo[0]*GRAD));
  C = 0.0;
  Q = 0.0;
  for (i=0; i<m; i++) {
    d = FeixeRetas[nImg1].Raio[i]-FeixeRetas[nImg2].Raio[i];
    sigma = FeixeRetas[nImg1].Sigma[i]*FeixeRetas[nImg1].Sigma[i]+
            FeixeRetas[nImg2].Sigma[i]*FeixeRetas[nImg2].Sigma[i];
    C+= 1.0/sigma;
    Q+= d/sigma;
  }
  s = fabs(Q/(Dp*C));
  *Ce = 1.0/(Dp*Dp*C);

  return s;
}

/***************************************************************************************************************
 MedPenMEvol
 -----------
 Aplica um processo de medição do tipo "evolutivo" à imagem da tela de uma peneira, identificando as retas
 definidoras de tramas e fios. Em seguida, obtém os fatores de escala e calcula os valores médios do
 diâmetro dos fios e da abertura das tramas verificando ainda as tolerâncias correspondentes.
 Parâmetros:
   nImg1  : imagem da tela da peneira
   nImg2  : imagem de retas localizadas em nImg1
   nImg3  : imagens com as retas localizadas em nImg1
   L      : comprimento mínimo das retas da imagem
   Esp    : espessura das retas da imagem
   angMin : limite inferior do intervalo de busca angular
   angMax : limite superior do intervalo de busca angular
   dAng   : incremento angular
   dPicos : distância mínima entre picos no espaço de Hough
   Fcorte : limiar de binarização do espaço de Hough
   nPc    : no. de picos a serem localizados no espaço de Hough
   RetPar : TRUE-> admite que as retas são paralelas
   abm    : abertura média da trama
   abmft  : TRUE se a abertura média estiver fora de tolerância
   pcabft1: % de aberturas maiores que o máximo tolerado
   pcabft2: % de aberturas na faixa admissível a no máximo 6%
   dm     : diâmetro médio dos fios
   pcdft  : % de diâmetros fora de tolerância
 ***************************************************************************************************************/
void MedPenMEvol(int nImg1,int nImg2,int nImg3,double L,double Esp,double angMin,double angMax,
     double dAng,int dPicos,double Fcorte,int nPc,BOOL RetPar,double *abm,BOOL *abmft,double *pcabft1,
     double *pcabft2,double *dm,double *pcdft)
{
  int sImg;             // tamanho da imagem
  int i;                // indice
  HGLOBAL hM1,hM2,hM3;  // manipuladores de memória
  LPBYTE imgBuf;        // buffer armazenando imagem original
  LPBYTE lpBits;        // imagem intermediária apresentada ao longo do processamento
  double limiar;        // limiar de binarização da imagem
  int np;               // número de pontos de um elemento estruturante
  ponto *ElemEst;       // elemento estruturante
  BOOL inicD1,inicD2;   // TRUE-> se as duas primeiras retas se referem a diâmetro de fio
  double er;            // erro da estimação dos fatores de escala

  // Alocação de memória para imagens auxiliares

  sImg = DimBitmap[nImg1].cx*DimBitmap[nImg1].cy;
  hM1 = GlobalAlloc(GHND, (DWORD)sImg);
  imgBuf = (LPBYTE)GlobalLock(hM1);
```

```
hM2 = GlobalAlloc(GHND, (DWORD)sImg);
lpBits = (LPBYTE)GlobalLock(hM2);
for (i=0; i<sImg; i++)
  imgBuf[i] = PixImg[nImg1][i];

// Limiariza a imagem da peneira

limiar = CalcLimiar(nImg1);
for (i=0; i<sImg; i++)
  PixImg[nImg1][i] = (PixImg[nImg1][i]<limiar) ? 0 : 255;

// Aplica sequência de operadores morfológicos de modo a obter uma imagem de bordas adequada

   // Passo 1: Eliminação de objetos que tocam as fronteiras da imagem
hM3 = GlobalAlloc(GHND, (DWORD)(100*sizeof(ponto)));
ElemEst = (ponto *) GlobalLock(hM3);
np = CriaElemEstCirc(1, ElemEst);
DestroiObjFront(nImg1, nImg2, np, ElemEst, 2, 6);
   // Passo 2: Fechamento de eventuais "buracos" da imagem
np = CriaElemEstCirc(2, ElemEst);
Fechamento(nImg2, nImg1, np, ElemEst);
   // Passo 3: Aplicação de detector de bordas morfológico à imagem
np = CriaElemEstCirc(1, ElemEst);
Erosao(nImg1, nImg2, np, ElemEst);
Dilatacao(nImg1, nImg3, np, ElemEst);
for (i=0; i<sImg; i++) {
  PixImg[nImg2][i]= (PixImg[nImg3][i]==PixImg[nImg2][i]) ? 0 : 255;
  lpBits[i] = PixImg[nImg2][i];
}
SetBitmapBits(hbm[nImg2], (DWORD)sImg, (LPSTR)lpBits);
sprintf(NomeImg[nImg2], "Bordas grossas da Imagem %d", nImg1);
MostraImagem(nImg2);
ApagaImagem(nImg1);
Esqueleto(nImg2, nImg1);

// Aplica a transformada de Hough para localizar as retas da imagem

DetCoordBordas(nImg1);
AplicaHough(nImg1,nImg2,FALSE,TRUE,TRUE,angMin,angMax,dAng,L,Esp,RetPar,dPicos,(nPc!=0),nPc,Fcorte);
AplicaHough(nImg1,nImg3,FALSE,TRUE,TRUE,angMin-92,angMax-88,dAng,L,Esp,RetPar,dPicos,(nPc!=0),nPc,Fcorte);

for (i=0; i<sImg; i++)
  PixImg[nImg1][i] = imgBuf[i];

// Associação das retas às medidas normalizadas de peneira

inicD1 = AssocMedNormPen(nImg2);
inicD2 = AssocMedNormPen(nImg3);

// Determinação de um modelo de medição de peneiras

ModMPen.ang1 = FeixeRetas[nImg2].Angulo[0];
ModMPen.a1 = 1.0/MapImgObj(nImg2,&er);
ModMPen.b1 = 0;
ModMPen.ang2 = FeixeRetas[nImg3].Angulo[0];
ModMPen.a2 = 1.0/MapImgObj(nImg3,&er);
ModMPen.b2 = 0;

// Determinação das distâncias no plano-objeto, utilizando-se o
// modelo de medição linear anterior

for (i=0; i<FeixeRetas[nImg2].nRetasImg; i++)
  FeixeRetas[nImg2].MedCal[i] = ModMPen.a1*(FeixeRetas[nImg2].Raio[i]-FeixeRetas[nImg2].Raio[0]);
FeixeRetas[nImg3].MedCal[0] = 0.0;
for (i=0; i<FeixeRetas[nImg3].nRetasImg; i++)
  FeixeRetas[nImg3].MedCal[i] = ModMPen.a2*(FeixeRetas[nImg3].Raio[i]-FeixeRetas[nImg3].Raio[0]);

// Medição da peneira pelo método sugerido em norma ABNT
// Critério 1: nenhum diâmetro de fio deve ultrapassar uma dada tolerância
// Critério 2: a abertura média deve ser limitada inferior e superiormente por uma tolerância dada
// Critério 3: 6% das aberturas não devem ultrapassar uma tolerância dada
// Critério 4: a máxima abertura não deve ultrapassar uma tolerância dada

MedicaodosFios(nImg2,inicD1,nImg3,inicD2,dm,pcdft);
MedicaodasTramas(nImg2,inicD1,nImg3,inicD2,abm,abmft,pcabft1,pcabft2);

// Liberação de memória

GlobalUnlock(hM1);
GlobalUnlock(hM2);
GlobalUnlock(hM3);

return;
}
```

```
/*****************************************************************************************************
 AssocMedNormPen
 ---------------
 Associa às retas localizadas na imagem de peneira as correspondentes medidas em mm, calculadas a partir
 de medidas-referência da abertura da trama e do diâmetro do fio.
 Parâmetros:
   nImg: número da imagem
 Retorna:
   TRUE: se as duas primeiras retas se referem a diâmetro de fio
 *****************************************************************************************************/
BOOL AssocMedNormPen(int nImg)
{
  double r1,r2;      // distâncias entre as retas 1&2 e 2&3
  double r;          // distância métrica de uma reta qualquer à origem
  double a,b;        // medida da abertura ou do diâmetro do fio
  int i;             // índice
  BOOL inic_Diam;    // TRUE-> duas primeiras retas se referem a diâmetro do fio

  r1 = FeixeRetas[nImg].Raio[1]-FeixeRetas[nImg].Raio[0];
  r2 = FeixeRetas[nImg].Raio[2]-FeixeRetas[nImg].Raio[1];
  if (r1<r2) {
    a = NPen.diam;
    b = NPen.abert;
    inic_Diam = TRUE;
  }
  else {
    a = NPen.abert;
    b = NPen.diam;
    inic_Diam = FALSE;
  }
  FeixeRetas[nImg].Medida[0] = 0.0;
  r = 0.0;
  for (i=1; i<FeixeRetas[nImg].nRetasImg; i++) {
    r = (i%2!=0) ? r+a : r+b;
    FeixeRetas[nImg].Medida[i] = r;
  }

  return(inic_Diam);
}

/*****************************************************************************************************
 MedicaodasTramas
 ----------------
 Calcula o valor médio das aberturas da peneira e calcula a porcentagem de tramas cujas aberturas se
 encontram fora da tolerância estabelecida em norma.
 Parâmetros:
   nImg1 : imagem com as retas localizadas na tela
   nImg2 : imagem com as retas localizadas na tela
   inicD1: índice inicial correspondente a fios na imagem 1
   inicD2: índice inicial correspondente a fios na imagem 2
   am    : abertura média das tramas
   aft   : TRUE se a abertura média estiver fora de tolerância
   pc1   : % de tramas com aberturas superiores à tolerância máxima
   pc2   : % de tramas com aberturas na faixa permitida a 6% das tramas
 *****************************************************************************************************/
void MedicaodasTramas(int nImg1, BOOL inicD1, int nImg2, BOOL inicD2, double *am, BOOL *aft, double *pc1,
     double *pc2)
{
  int i,j,k,m,n;             // índices
  int k1;                    // no. de tramas com abertura superior à máxima permitida
  int k2;                    // no. de tramas com abertura na faixa de tolerância para apenas 6% das tramas
  int n1,n2;                 // número de retas nas imagens 1 e 2
  double a1,a2;              // aberturas de uma trama em duas direções ortogonais
  double X[4],Y[4];          // intersecções de retas
  double a;                  // abertura média de uma trama
  double *R1, *R2;           // distâncias das retas nas imagens 1 e 2 medidas a partir da origem
  HGLOBAL hM1,hM2,hM3,hM4;   // manipuladores de memória
  double *Ang1,*Ang2;        // orientação das retas nas imagens 1 e 2 relativamente ao eixo X

  // Salva as distâncias e os ângulos dos feixes de retas das imagens 1 e 2

  n1 = FeixeRetas[nImg1].nRetasImg;
  n2 = FeixeRetas[nImg2].nRetasImg;
  hM1  = GlobalAlloc(GHND, (DWORD)(n1*sizeof(double)));
  R1   = (double *) GlobalLock(hM1);
  hM2  = GlobalAlloc(GHND, (DWORD)(n2*sizeof(double)));
  R2   = (double *) GlobalLock(hM2);
  hM3  = GlobalAlloc(GHND, (DWORD)(n1*sizeof(double)));
  Ang1 = (double *) GlobalLock(hM3);
  hM4  = GlobalAlloc(GHND, (DWORD)(n2*sizeof(double)));
  Ang2 = (double *) GlobalLock(hM4);
  for (i=0; i<n1; i++) {
    R1[i] = FeixeRetas[nImg1].Raio[i];
```

```
    Ang1[i] = FeixeRetas[nImg1].Angulo[i];
  }
  for (i=0; i<n2; i++) {
    R2[i] = FeixeRetas[nImg2].Raio[i];
    Ang2[i] = FeixeRetas[nImg2].Angulo[i];
  }

  // Ajusta os índices iniciais correspondentes a tramas nas imagens 1 e 2

  i = (inicD1) ? 1 : 0;
  j = i+1;
  m = (inicD2) ? 1 : 0;
  n = m+1;
  k = 0;
  k1 = 0;
  k2 = 0;
  *am = 0.0;
  FeixeRetas[nImg1].nRetasImg = 2;
  FeixeRetas[nImg2].nRetasImg = 2;

  // Determina os valores médios de abertura para cada trama e conta as que fogem à tolerância dada

  while (j<n1) {
    FeixeRetas[nImg1].Angulo[0] = Ang1[i];
    FeixeRetas[nImg1].Angulo[1] = Ang1[j];
    FeixeRetas[nImg1].Raio[0] = R1[i];
    FeixeRetas[nImg1].Raio[1] = R1[j];
    while (n<n2) {
      FeixeRetas[nImg2].Angulo[0] = Ang2[m];
      FeixeRetas[nImg2].Angulo[1] = Ang2[n];
      FeixeRetas[nImg2].Raio[0] = R2[m];
      FeixeRetas[nImg2].Raio[1] = R2[n];
      if (InterFeixes(nImg1, nImg2, X, Y)==4) { // as 4 intersecções são interiores à moldura
        a1 = FeixeRetas[nImg1].MedCal[j]-FeixeRetas[nImg1].MedCal[i];
        a2 = FeixeRetas[nImg2].MedCal[n]-FeixeRetas[nImg2].MedCal[m];
        a  = (a1+a2)*0.5;
        *am+= a;
        k++;
        if ((a1>NPen.abert+NPen.tmax_abert) || (a2>NPen.abert+NPen.tmax_abert))
          k1++;
        else if (((a1>NPen.abert+NPen.t6_abert) && (a1<NPen.abert+NPen.tmax_abert)) ||
                 ((a2>NPen.abert+NPen.t6_abert) && (a2<NPen.abert+NPen.tmax_abert)))
          k2++;
      }
      m+= 2;
      n+= 2;
    }
    i+= 2;
    j+= 2;
  }
  *am  = *am/(double)k;
  *pc1 = (double)k1/(double)k;
  *pc2 = (double)k2/(double)k;
  *aft = ((*am<NPen.abert-NPen.tm_abert) || (*am>NPen.abert+NPen.tm_abert)) ? TRUE : FALSE;

  // Recupera os números de retas das imagens 1 e 2

  FeixeRetas[nImg1].nRetasImg = n1;
  for (i=0; i<n1; i++) {
    FeixeRetas[nImg1].Angulo[i] = Ang1[i];
    FeixeRetas[nImg1].Raio[i] = R1[i];
  }
  FeixeRetas[nImg2].nRetasImg = n2;
  for (i=0; i<n2; i++) {
    FeixeRetas[nImg2].Angulo[i] = Ang2[i];
    FeixeRetas[nImg2].Raio[i] = R2[i];
  }

  GlobalUnlock(hM1);
  GlobalUnlock(hM2);
  GlobalUnlock(hM3);
  GlobalUnlock(hM4);

  return;
}

/*****************************************************************************************************************
 MedicaodosFios
 --------------
 Calcula o valor médio dos fios da peneira e conta o número de fios que se encontram fora da tolerância
 estabelecida em norma.
 Parâmetros:
   nImg1 : imagem com as retas localizadas na peneira
   nImg2 : imagem com as retas localizadas na peneira
```

```
   inicD1: índice inicial correspondente a "fios" na imagem 1
   inicD2: índice inicial correspondente a "fios" na imagem 2
   dm    : diâmetro médio dos fios
   pcft  : porcentagem de fios com diâmetros fora de tolerância
 ************************************************************************************************************/
void MedicaodosFios(int nImg1, BOOL inicD1, int nImg2, BOOL inicD2, double *dm, double *pcft)
{
  int i,j,k;     // índices
  int n1,n2;     // número de retas nas imagens 1 e 2
  double d;      // diâmetro de fio
  double l;      // comprimento de fio
  double L;      // comprimento total dos fios
  int k1;        // número de fios fora de tolerância

  // Mede os diâmetros na imagem nImg1 (Theta)

  n1 = FeixeRetas[nImg1].nRetasImg;
  i = (inicD1) ? 0 : 1;
  j = i+1;
  *dm = 0.0;
  L = 0.0;
  k = 0;
  k1 = 0;
  while (j<n1) {
    d = FeixeRetas[nImg1].MedCal[j]-FeixeRetas[nImg1].MedCal[i];
    l = (double)(CalcCompReta(nImg1,i)+CalcCompReta(nImg1,j))*0.5;
    if ((d>NPen.diamax) || (d<NPen.diamin))
      k1++;
    *dm+= d*l;
    L+= l;
    i+=2;
    j+=2;
    k++;
  }

  // Mede os diâmetros na imagem nImg2 (Theta+90graus)

  n2 = FeixeRetas[nImg2].nRetasImg;
  i = (inicD2) ? 0 : 1;
  j = i+1;
  while (j<n1) {
    d= FeixeRetas[nImg1].MedCal[j]-FeixeRetas[nImg1].MedCal[i];
    l = (double)(CalcCompReta(nImg2,i)+CalcCompReta(nImg2,j))*0.5;
    if ((d>NPen.diamax) || (d<NPen.diamin))
      k1++;
    *dm+= d*l;
    L+= l;
    i+=2;
    j+=2;
    k++;
  }
  *dm = *dm/L;
  *pcft = (double)k1/(double)k;

  return;
}
/************************************************************************************************************
 MedPenMMorf
 -----------
 Utiliza filtros morfológicos para selecionar aberturas de tramas e diâmetros de fios de peneiras dentro
 das faixas de tolerância requeridas pela norma ABNT de Peneiras de Medição.
 Parâmetros:
   nImg1    : imagem da peneira
   nImg2    : imagem com objetos relacionados com as tramas
   nImg3    : imagem com objetos relacionados com os fios
   apTramas: TRUE-> mede apenas as aberturas da peneira
   usTD     : TRUE-> calcula o diâmetro médio usando Transformada Distância
   abm      : abertura média da trama
   abmft    : TRUE se a abertura média estiver fora de tolerância
   pcab1    : % de aberturas maiores que o mínimo recomendado
   pcab2    : % de aberturas no intervalo 1 (vide norma ABNT)
   pcab3    : % de aberturas no intervalo 2 (vide norma ABNT)
   pcab4    : % de aberturas no intervalo 3 (vide norma ABNT)
   pcab5    : % de aberturas maiores que o máximo admissível
   dm       : diâmetro médio dos fios
 ************************************************************************************************************/
void MedPenMMorf(int nImg1,int nImg2,int nImg3,BOOL apTramas,BOOL usTD,double *abm,BOOL *abmft,
                 double *pcab1,double *pcab2,double *pcab3,double *pcab4,double *pcab5,double *dm)
{
  int sImg;                  // tamanho da imagem
  int i;                     // indice
  HGLOBAL hM1,hM2;           // manipuladores de memória
  LPBYTE imgBuf1,imgBuf2;    // imagens auxiliares
  double limiar;             // limiar de binarização da imagem
```

```
int np;                    // número de pontos de um elemento estruturante
ponto ElemEst[200];        // elemento estruturante
int ntramas;               // no. de "tramas completas" na imagem da peneira
int ntrab1;                // no. de tramas com abertura maior que abertura_nominal+tmax
int ntrab2;                // no. de tramas com abertura menor que abertura_nominal-tm
int ntrab3;                // no. de tramas com abertura no intervalo [abertura_nominal +/- tm]
int ntrab4;                // no. de tramas com abertura em [abertura_nominal+t6,abertura_nominal+tmax]
int ntrab5;                // no. de tramas com abertura em [abertura_nominal+tm,abertura_nominal+t6]
double r1,r2;              // abertura nominal medida em número de pixels segundo direções ortogonais
double area;               // área média das tramas em número de pixels ao quadrado
double d1,d2;              // diâmetros dos fios em número de pixels, segundo direções ortogonais
double D1,D2;              // diâmetros dos fios em mm, segundo direções ortogonais
char str[100];             // string

// Alocação de memória para imagens auxiliares

sImg = DimBitmap[nImg1].cy*DimBitmap[nImg1].cx;
hM1 = GlobalAlloc(GHND, (DWORD)sImg);
imgBuf1 = (LPBYTE)GlobalLock(hM1);
hM2 = GlobalAlloc(GHND, (DWORD)sImg);
imgBuf2 = (LPBYTE)GlobalLock(hM2);
for (i=0; i<sImg; i++)
  imgBuf1[i] = PixImg[nImg1][i];

// Limiarização  da imagem da peneira

limiar = CalcLimiar(nImg1);
for (i=0; i<sImg; i++)
  PixImg[nImg1][i] = (PixImg[nImg1][i]<limiar) ? 0 : 255;

// Sequência de operações morfológicas para medição da abertura da trama

   // Passo 1: Regularização da imagem limiarizada

np = CriaElemEstCirc(1, ElemEst);
DestroiObjFront(nImg1, nImg2, np, ElemEst, 2, 6);
for (i=0; i<sImg; i++)
  PixImg[nImg1][i] = PixImg[nImg2][i];
np = CriaElemEstCirc(2, ElemEst);
Fechamento(nImg1, nImg2, np, ElemEst);
sprintf(NomeImg[nImg2], "Imagem binarizada e regularizada de peneira");
SetBitmapBits(hbm[nImg2], (DWORD)sImg, (LPSTR)PixImg[nImg2]);
MostraImagem(nImg2);

  // Passo 2: Estimação da abertura média das tramas

for (i=0; i<sImg; i++) {
  PixImg[nImg1][i] = PixImg[nImg2][i];
  // Salva imagem limiarizada e regularizada
  imgBuf2[i] = PixImg[nImg1][i];
}
area = 0.0;
for (i=0; i<sImg; i++)
  if (PixImg[nImg1][i]==255)
    area+= 1.0;
ntramas = Rotulacao(nImg1, nImg2, 0.0);
area = area/(double)ntramas;
*abm = sqrt(area*ModMPen.a1*ModMPen.a2);
*abmft = ((*abm>=NPen.abert-NPen.tm_abert) && (*abm<=NPen.abert+NPen.tm_abert))?FALSE:TRUE;

  // Passo 3: Verificação de tolerâncias

// % de aberturas > a+tmax
r1 = 0.5*(NPen.abert+NPen.tmax_abert)/ModMPen.a1;
r2 = 0.5*(NPen.abert+NPen.tmax_abert)/ModMPen.a2;
sprintf(str, "aberturas > %6.4f", NPen.abert+NPen.tmax_abert);
ntrab1 = Sonda(nImg1, nImg2, imgBuf2, r1, r2, str);
*pcab5 = (double)ntrab1/(double)ntramas;

// % de aberturas < a-tm
r1 = 0.5*(NPen.abert-NPen.tm_abert)/ModMPen.a1;
r2 = 0.5*(NPen.abert-NPen.tm_abert)/ModMPen.a2;
sprintf(str, "aberturas > %6.4f", NPen.abert-NPen.tm_abert);
ntrab2 = ntramas - Sonda(nImg1, nImg2, imgBuf2, r1, r2, str);
*pcab1 = (double)ntrab2/(double)ntramas;

// % de aberturas > a-tm e < a+tm
r1 = 0.5*(NPen.abert+NPen.tm_abert)/ModMPen.a1;
r2 = 0.5*(NPen.abert+NPen.tm_abert)/ModMPen.a2;
sprintf(str, "aberturas >%6.4f", NPen.abert+NPen.tm_abert);
ntrab3 = ntramas-ntrab2-Sonda(nImg1, nImg2, imgBuf2, r1, r2, str);
*pcab2 = (double)ntrab3/(double)ntramas;

// % de aberturas > a+tm e < a+t6
```

```
// % de aberturas > a+t6 e < a+tmax
r1 = 0.5*(NPen.abert+NPen.t6_abert)/ModMPen.a1;
r2 = 0.5*(NPen.abert+NPen.t6_abert)/ModMPen.a2;
sprintf(str, "aberturas >%6.4f", NPen.abert+NPen.t6_abert);
ntrab4 = Sonda(nImg1, nImg2, imgBuf2, r1, r2, str)-ntrab1;
*pcab4 = (double)ntrab4/(double)ntramas;
ntrab5 = ntramas-ntrab2-ntrab3-ntrab4-ntrab1;
*pcab3 = (double)ntrab5/(double)ntramas;

if (apTramas) {
  GlobalUnlock(hM1);
  GlobalUnlock(hM2);
  return;
}

// Sequência de operações morfológicas para medição do diâmetro do fio

  // Passo 1: Separação dos objetos "fios" em duas imagens

for (i=0; i<sImg; i++)
  PixImg[nImg1][i] = abs(imgBuf2[i]-255);
r1 = 0.8*NPen.diam/ModMPen.a1;
SeparaFios(nImg1, nImg2, r1, ModMPen.ang1);
r2 = 0.8*NPen.diam/ModMPen.a2;
for (i=0; i<sImg; i++)
  PixImg[nImg1][i] = abs(imgBuf2[i]-255);
SeparaFios(nImg1, nImg3, r2, ModMPen.ang2);

  // Passo 2: Determinação do diâmetro dos fios

r1 = floor(0.5*NPen.diamin/ModMPen.a1)-1.0;
r2 = floor(0.5*NPen.diamin/ModMPen.a2)-1.0;
if (!usTD) {
  // Método baseado no esqueleto das imagens
  for (i=0; i<sImg; i++)
    PixImg[nImg1][i] = PixImg[nImg2][i];
  d1 = DiamMMorf(nImg1, nImg2, 20, r1);
  for (i=0; i<sImg; i++)
    PixImg[nImg1][i] = PixImg[nImg3][i];
  d2= DiamMMorf(nImg1, nImg3, 20, r2);
}
else {
  // Método baseado na Transformada-Distância
  for (i=0; i<sImg; i++)
    PixImg[nImg1][i] = PixImg[nImg2][i];
  d1 = DiamMMorf2(nImg1, nImg2, (int)r1);
  for (i=0; i<sImg; i++)
    PixImg[nImg1][i] = PixImg[nImg3][i];
  d2 = DiamMMorf2(nImg1, nImg3, (int)r2);
}
D1 = ModMPen.a1*d1;
D2 = ModMPen.a2*d2;
*dm = (D1+D2)/2.0;

// Recuperação da imagem original

for (i=0; i<sImg; i++)
  PixImg[nImg1][i] = imgBuf1[i];

// Liberação de memória

GlobalUnlock(hM1);
GlobalUnlock(hM2);

return;
}

/*****************************************************************************************************
 Sonda
 -----
 Erode imagem com elemento estruturante composto por um ponto central e 4 pontos radiais a distâncias do
 centro r1 e r2, de modo que os objetos remanescentes são os que possuem tamanho superior a um valor dado.
 Parâmetros:
   nImg1 : imagem original
   nImg2 : imagem com os objetos remanescentes da erosão
   imgBuf: pixels da imagem nImg1
   r1,r2 : tolerâncias (em pixels) segundo direções ortogonais
   tit   : título -> tipo de verificação
 *****************************************************************************************************/
int Sonda(int nImg1, int nImg2, LPBYTE imgBuf, double r1, double r2, char *tit)
{
  int sImg;              // tamanho da imagem
  double ang1,ang2;      // orientações das retas médias definidoras das tramas da peneira
  ponto ElemEst[100];    // elemento estruturante verificador de tolerâncias
```

```
  int i;                // índice
  int np;               // número de pontos de elemento estruturante

  sImg = DimBitmap[nImg1].cx*DimBitmap[nImg1].cy;
  ang1 = fabs(ModMPen.ang1*GRAD);
  ang2 = fabs(ModMPen.ang2*GRAD);
  for (i=0; i<sImg; i++)
    PixImg[nImg1][i] = imgBuf[i];

  // Processo de verificação de tolerância:
  // --------------------------------------
  // Erode imagem com elemento estruturante definido por um ponto central e 4
  // pontos radiais a distâncias r1=tol1 e r2=tol2 do centro, de modo que  os
  // objetos remanescentes são os que possuem tamanho superior à tolerância dada.

  CriaElemEst5Pts(r1,ang1,r2,ang2,ElemEst);
  Erosao(nImg1, nImg2, 5, ElemEst);
  for (i=0; i<sImg; i++)
    PixImg[nImg1][i] = PixImg[nImg2][i];

  np = CriaElemEstCirc(2, ElemEst);
  Dilatacao(nImg1, nImg2, np, ElemEst);

  SetBitmapBits(hbm[nImg2], (DWORD)sImg, (LPSTR)PixImg[nImg2]);
  sprintf(NomeImg[nImg2], tit);
  MostraImagem(nImg2);

  for (i=0; i<sImg; i++)
    PixImg[nImg1][i] = PixImg[nImg2][i];

  return Rotulacao(nImg1, nImg2, 0.0);
}

/*****************************************************************************************************
 SeparaFios
 ----------
 Erode imagem com elemento estruturante composto por um ponto central e 2 pontos radiais a distâncias dadas
 do centro e definindo uma reta segundo um ângulo dado com a vertical.
 Parâmetros:
   nImg1 : imagem original
   nImg2 : imagem com os objetos remanescentes da erosão
   r     : distância (em pixels) do ponto ao centro
   ang   : angulo da reta com a vertical
 *****************************************************************************************************/
void SeparaFios(int nImg1, int nImg2, double r, double ang)
{
  int nLin, nCol;       // número de linhas e de colunas da imagem
  int sImg;             // tamanho da imagem
  ponto ElemEst[25];    // elemento estruturante verificador de dimensão
  int np;               // número de pontos de elemento estruturante
  int nfios;            // número de fios maiores que 0.20 x o comprimento do maior fio da imagem
  int i,j,k;            // índices

  nLin = DimBitmap[nImg1].cy;
  nCol = DimBitmap[nImg1].cx;
  sImg = DimBitmap[nImg1].cx*DimBitmap[nImg1].cy;

  // Erode imagem com elemento estruturante composto por um ponto central e 2 pontos radiais
  // a uma distância r do centro e definindo uma reta formando um ângulo ang com a vertical

  for (i=0; i<nLin; i++)
    for (j=0; j<nCol; j++) {
      k = i*nCol+j;
      if ((i<50) || (i>nLin-50) || (j<50) || (j>nCol-50))
        PixImg[nImg1][k] = 0;
    }
  CriaElemEst3Pts(r,ang*GRAD,ElemEst);
  Erosao(nImg1, nImg2, 3, ElemEst);
  np = CriaElemEstCirc(2, ElemEst);
  Abertura(nImg2, nImg2, np, ElemEst);
  SetBitmapBits(hbm[nImg2], (DWORD)sImg, (LPSTR)PixImg[nImg2]);
  sprintf(NomeImg[nImg2], "Fios a %5.1f graus com a vertical", ang);
  MostraImagem(nImg2);
  for (i=0; i<sImg; i++) {
    PixImg[nImg1][i] = PixImg[nImg2][i];
    PixImg[nImg2][i] = 0;
  }
  nfios = Rotulacao(nImg1, nImg2, 0.2);

  return;
}

/*****************************************************************************************************
 DiamMMorf
```

```
 ---------
 Determina o diâmetro médio dos objetos "fios" da imagem da peneira através de método morfológico baseado
 no perímetro do esqueleto da imagem.
 Parâmetros:
   nImg1: imagem original
   nImg2: imagem de trabalho e imagem final
   marg : margem adotada para evitar os efeitos de borda
   r    : menor raio previsto para os objetos "fios"
 ***************************************************************************************************************/
double DiamMMorf(int nImg1, int nImg2, int marg, double r)
{
  int nCol, nLin;        // número de linhas e de colunas da imagem
  int sImg;              // tamanho da imagem
  int i,j,k;             // índices
  double area;           // área da imagem
  double L;              // comprimento do esqueleto da imagem
  int np;                // número de pontos do elemento estruturante
  ponto ElemEst[300];    // elemento estruturante

  // Aplica à imagem uma erosão que, com certeza, não altera a conectividade
  // original dos objetos. Em seguida, calcula a área dessa imagem.

  nCol = DimBitmap[nImg1].cx;
  nLin = DimBitmap[nImg1].cy;
  sImg = nLin*nCol;
  np = CriaElemEstAro((int)r,ElemEst);
  Erosao(nImg1, nImg2, np, ElemEst);
  area = 0.0;
  for (i=marg; i<nLin-marg; i++)
    for (j=marg; j<nCol-marg; j++) {
      k = i*nCol+j;
      if (PixImg[nImg2][k] == 255)
        area+= 1.0;
    }
  for (i=0; i<sImg; i++)
    PixImg[nImg1][i] = PixImg[nImg2][i];

  // Esqueletoniza a imagem erodida anterior e determina o perímetro do esqueleto

  Esqueleto(nImg1, nImg2);
  L=0.0;
  for (i=marg; i<nLin-marg; i++)
    for (j=marg; j<nCol-marg; j++) {
      k = i*nCol+j;
      if (PixImg[nImg2][k]==255)
        L+= 1.0;
    }

  // O diâmetro médio dos objetos da imagem é dado, por: d = 2*r+Area/L,
  // onde r é o raio do elemento estruturante
  //      Area é a área da imagem erodida
  //      L é o perímetro do esqueleto da imagem erodida

  return(2*r+area/L);
}

/***************************************************************************************************************
 DiamMMorf2
 ----------
 Determina o diâmetro médio dos fios da imagem da peneira através de método morfológico baseado na
 Transformada-Distância da imagem.
 Parâmetros:
   nImg1: imagem original
   nImg2: imagem de trabalho e imagem final
   r    : raio nominal mínimo esperado
 Retorna:
   diâmetro estimado
 ***************************************************************************************************************/
double DiamMMorf2(int nImg1, int nImg2, int r)
{
  int sImg;              // tamanho da imagem
  int i,n;               // índices
  int np;                // número de pontos do elemento estruturante
  ponto ElemEst[25];     // elemento estruturante
  int MaxD;              // distância máxima identificada na imagem pela Transformada Distância
  HGLOBAL hM1,hM2,hM3;   // manipuladores de memória
  LPBYTE lpBits;         // pixels da imagem
  float *D;              // transformada-distância da imagem
  double *DM;            // máximos locais da transformada-distância
  double dm;             // diâmetro médio dos objetos localizados na imagem nImg1

  // Alocação de memória

  sImg = DimBitmap[nImg1].cx*DimBitmap[nImg1].cy;
```

```
  hM1 = GlobalAlloc(GHND, (DWORD)sImg);
  lpBits = (LPBYTE) GlobalLock(hM1);
  hM2 = GlobalAlloc(GHND, (DWORD)(sImg*sizeof(float)));
  D = (float *) GlobalLock(hM2);
  hM3 = GlobalAlloc(GHND, (DWORD)(sImg*sizeof(double)));
  DM = (double *) GlobalLock(hM3);

  // Geração da Transformada-Distância

  ApagaImagem(nImg2);
  np = CriaElemEstCirc(1, ElemEst);
  MaxD = TransfDist(nImg1, nImg2, np, ElemEst);
  for (i=0; i<sImg; i++) {
    D[i] = PixImg[nImg2][i]*(float)MaxD/255.0F;
    lpBits[i] = ConvIndPal[PixImg[nImg2][i]];
  }
  SetBitmapBits(hbm[nImg2], (DWORD)sImg, (LPSTR)lpBits);
  sprintf(NomeImg[nImg2], "Transformada Distância da imagem");
  MostraImagem(nImg2);
  GlobalUnlock(hM1);

  // Binarização e Esqueletonização da imagem da Transformada-Distância

  for (i=0; i<sImg; i++) {
    PixImg[nImg1][i] =  (PixImg[nImg2][i]>=(r-1)*255.0F/(float)MaxD) ? 255 : 0;
    PixImg[nImg2][i] = 0;
  }
  Esqueleto(nImg1, nImg2);
  n=0;
  for (i=0; i<sImg; i++)
    if (PixImg[nImg2][i]==255) {
      DM[n] = (double)D[i];
      n++;
    }
  GlobalUnlock(hM2);

  // Determinação do valor mediano dos máximos locais da Transformada-Distância

  sort(n,DM);
  dm = 2*DM[n/2];
  GlobalUnlock(hM3);

  return(dm);
}

/*****************************************************************************************************
 MEDICAO.H
 ---------

 Definições e declarações de tipos e variáveis utilizadas pelo módulo MEDICAO.CPP
 *****************************************************************************************************/

#define STRICT
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

/*****************************************************************************************************
  DEFINIÇÕES
 *****************************************************************************************************/

#define MAXSTRINGSIZE 150                          // tamanho máximo de string
#define NMAX_IMG 20                                // número máximo de imagens alocadas
#define N_PixelsX 640                              // número máximo de colunas da imagem
#define N_PixelsY 480                              // número máximo de linhas da imagem
#define NMAX_RETAS_IMG 100                         // número máximo de retas em imagem

/*****************************************************************************************************
 VARIÁVEIS UTILIZADAS POR OBJETOS GRÁFICOS DA APLICAÇÃO
 *****************************************************************************************************/

extern HBITMAP hbm[NMAX_IMG];                      // manipuladores de imagem
extern BYTE ConvIndPal[256];                       // tabela de conversão de índices de cor

/*****************************************************************************************************
 VARIÁVEIS ASSOCIADAS A IMAGENS
 *****************************************************************************************************/

extern BYTE PixImg[NMAX_IMG][N_PixelsX*N_PixelsY]; // imagens
extern SIZE DimBitmap[NMAX_IMG];                   // dimensões das imagens
extern char NomeImg[NMAX_IMG][MAXSTRINGSIZE];      // nomes das imagens
extern int nImgReg;                                // número da última imagem gravada
```

```
extern double SigmaN;                                   // desvio-padrão do ruído Gaussiano das imagens
extern struct {
  float XBorda[4*N_PixelsX*N_PixelsY];                  // abcissas das bordas
  float YBorda[4*N_PixelsX*N_PixelsY];                  // ordenadas das bordas
} CoordBordas;                                          // coordenadas das bordas da última imagem de bordas
extern BYTE imgRec[4*N_PixelsX*N_PixelsY];              // última imagem reconstruída

/*****************************************************************************************************
 VARIÁVEIS ASSOCIADAS A FEIXES DE RETAS
 *****************************************************************************************************/

extern struct {
  int nRetasImg;                                        // número de retas da imagem
  BOOL tipo;                                            // tipo de feixe-> TRUE: linear ; FALSE: angular
  double Raio  [NMAX_RETAS_IMG];                        // distância da reta medida a partir da origem(0,0)
  double Angulo[NMAX_RETAS_IMG];                        // orientação da reta relativamente ao eixo X
  double Medida[NMAX_RETAS_IMG];                        // medida linear ou angular da escala-objeto associada
  double MedCal[NMAX_RETAS_IMG];                        // medida linear ou angular calculada
  double Sigma [NMAX_RETAS_IMG];                        // desvio-padrão da reta média ajustada
  int    nPts  [NMAX_RETAS_IMG];                        // número de pontos da reta
} FeixeRetas[NMAX_IMG];                                 // feixe de retas da imagem

/*****************************************************************************************************
 VARIÁVEIS PARA CALIBRAÇÃO DE MICROESCALAS
 *****************************************************************************************************/

extern struct {
  double ah;                                            // Horizontal: parâmetro "a" (não utilizado)
  double bh;                                            // Horizontal: parâmetro "b"
  double ch;                                            // Horizontal: parâmetro "c" (não utilizado)
  double av;                                            // Vertical  : parâmetro "a" (não utilizado)
  double bv;                                            // Vertical  : parâmetro "b"
  double cv;                                            // Vertical  : parâmetro "c" (não utilizado)
} ModMed;                                               // fatores de escala nas direções horizontal e vertical

/*****************************************************************************************************
 VARIÁVEIS PARA CALIBRAÇÃO DE PENEIRAS
 *****************************************************************************************************/

extern struct {
  double abert;                                         // abertura da peneira
  double tm_abert;                                      // tolerância média da abertura
  double t6_abert;                                      // tolerância máxima para 6% das aberturas
  double tmax_abert;                                    // tolerância máxima de uma abertura
  double diam;                                          // diâmetro dos fios
  double diamin;                                        // diâmetro mínimo
  double diamax;                                        // diâmetro máximo
} NPen;                                                 // norma dimensional ABNT de peneiras
extern struct {
  double ang1;                                          // ângulo médio do primeiro feixe de retas
  double a1;                                            // coeficiente linear
  double b1;                                            // coeficiente constante
  double ang2;                                          // ângulo médio do segundo feixe de retas
  double a2;                                            // coeficiente linear
  double b2;                                            // coeficiente constante
} ModMPen;                                              // modelo de medição de peneira

/*****************************************************************************************************
 TIPOS E VARIÁVEIS DE USO GERAL
 *****************************************************************************************************/

typedef struct {int x; int y;} ponto;                   // coordenadas (x,y) de um ponto
BOOL MedemArq;                                          // TRUE -> medidas reais em arquivo
extern double PI;                                       // Pi
extern double GRAD;                                     // conversão de graus para radianos

/*****************************************************************************************************
 FUNÇÕES DO MÓDULO MEDIÇÃO.H
 *****************************************************************************************************/

void MedEscFreqEsp(int nImgIn, int nImgOut, BOOL DirV, double distT);
void MedEscTHough(int nImgIn,int nImgOut,double AngMin,double AngMax,double DelAng,double Compri,double Esp,
     int dPicos, double Fcorte, BOOL dHor);
void ConvnPixelsmm(int nImg, BOOL dHor);
void MedEscMEvol1(int nImg1,int nImg2,int nImg3,int limiar,BOOL iEsq,BOOL ordTHough,double L,double Esp,
     double AngMin,double AngMax,double DelAng,int distPicos,double Fcorte,BOOL racMMQ,BOOL iSig1,BOOL dHor);
void MedEscMEvol2(int nImg1,int nImg2,int nImg3,int W,int tDet,BOOL iReconst,int tBorda,double L,double Esp,
     double AngMin,double AngMax,double DelAng,int distPicos,double Fcorte,BOOL racMMQ,BOOL iSig1,BOOL dHor,
     BOOL estN);
void MedEscMEvol3(int nImg1, int nImg2, int nImg3, BOOL DetPtMin, double L, double Esp, double AngMin,
     double AngMax, double DelAng, int distPicos, double Fcorte, BOOL iSig1, BOOL dHor);
void MedEscpDesloc(int nImg1, int nImg2, int nImg3, int nImg4, BOOL escRMed, double distT, double delta,
     BOOL dHor, double angMin, double angMax, double Fcorte, int tDet, BOOL fMedia, int W);
void MedPenMEvol(int nImg1,int nImg2,int nImg3,double L,double Esp,double angMin,double angMax,double dAng,
```

```
    int dPicos,double Fcorte,int nPc,BOOL RetPar,double *abm,BOOL *abmft,double *pcabft1, double *pcabft2,
    double *dm,double *pcdft);
void MedPenMMorf(int nImg1,int nImg2,int nImg3,BOOL apTramas,BOOL usEsq,double *abm,BOOL *abmft,
    double *pcab1,double *pcab2,double *pcab3,double *pcab4,double *pcab5,double *dm);
void SeqOpCalibracao(int nImg1, int nImg2, int nImg3, int nImg4, BOOL escRMed, double distT, BOOL fMedia,
    int W, int tDet, double L, double Esp, double angMin, double angMax, double DelAng, int distPicos,
    double Fcorte, double delta, BOOL dHor);
void SeqAjusteRetas(int nImg1, int nImg2, int nImg3, int tDet, int tBorda, double pcfBor, double angMin,
    double angMax, double dAng, double L, double Esp, int dPicos, double Fcorte, BOOL AjRet, BOOL Rec,
    BOOL cEsRef);
void GeraIndPxViz(BOOL dHor,int *lin1,int *lin2,int *lin3,int *lin4,int *col1,int *col2,int *col3,int *col4,
    float *fx,float *fy);
void DetTracoCorr(LPBYTE imgEsc, int nImgTr, int nImgRt, int nImgF, BOOL dHor, int ntraco, float *Ydtr, float
xmin);
void DetTracoPMin(LPBYTE imgEsc, int nImgTr, int nImgRt, int nImgF, BOOL dHor, int ntraco, float *Ydtr);
void ConvUnidEsp(LPBYTE img,float *M,float xmin,float fx,float fy,int a,int b,int c,int d,int e,int f,int g,
    int n,float *p,float *x,float *y);
void MedicaodosFios(int nImg1, BOOL inicD1, int nImg2, BOOL inicD2, double *dm, double *pc1);
void MedicaodasTramas(int nImg1, BOOL inicD1, int nImg2, BOOL inicD2, double *am, BOOL *aft, double *pc1,
    double *pc2);
void SeparaFios(int nImg1, int nImg2, double r, double ang);
BOOL AssocMedNormPen(int nImg);
int Sonda(int nImg1, int nImg2, LPBYTE imgBuf, double r1, double r2, char *tit);
float GeraDetTraco(LPBYTE imgEsc, int nImgTr, int nImgRt, int ntraco, BOOL dHor, float *Ydtr);
double MapImgObj(int nImg, double *Ce);
double EstimaFEsc(int nImg1, int nImg2, double D, double *Ce);
double DiamMMorf(int nImg1, int nImg2, int marg, double r);
double DiamMMorf2(int nImg1, int nImg2, int r);

/*****************************************************************************************************
 FUNÇÕES EXTERNAS
 *****************************************************************************************************/

extern void SalvaImgAuto(int nImg);
extern void ApagaImagem(int nImg);
extern void DetCoordBordas(int nImgIn);
extern void FiltroMedia(int nImgIn, int nImgOut, int largJan);
extern void FiltroMediana(int nImgIn, int nImgOut, int largJan);
extern void Reconstrucao(int nImgIn, int nImgOut);
extern void DBordasRoberts(int nImgIn, int nImgOut, int tipBorda, BOOL Bin, double pcFBordas, BOOL Rec);
extern void DBordasSobel(int nImgIn, int nImgOut, int tipBorda, BOOL Bin, double pcFBordas, BOOL Rec);
extern void DBordasSubPixel5x5(int nImgIn, int nImgOut, int tipBorda, BOOL Bin, double pcFBordas, BOOL Rec);
extern void DBordasSubPixel3x3(int nImgIn, int nImgOut, int tipBorda, BOOL Bin, double pcFBordas, BOOL Rec);
extern void MostraImagem(int nImg);
extern void AplicaHough(int nImgIn,int nImgOut,BOOL Rec,BOOL iBin,BOOL aut,double AngMin,double AngMax,
       double DelAng,double Compri,double MEsp,BOOL RetPar,int distPicos,BOOL nfxPicos,int nPicos,
       double Fcorte);
extern void AjustaRetas1(int nImgBor, int nImgRtAp, int nImgRtAj, double largJan, BOOL Dup);
extern void AjustaRetas2(int nImgBor, int nImgRtAp, int nImgRtAj, double largJan, BOOL Dup);
extern void InfoCaractImg(int nImg, int caso, int nc);
extern void AssociaImagemObjeto(int nImg, BOOL fLinear, BOOL medReal);
extern void FeixeMedio(int nImg1, int nImg2, int nImgOut);
extern void Esqueleto(int nImgIn, int nImgOut);
extern void ludcmp(double **a, int n, int *indx, double *d);
extern void lubksb(double **a, int n,int *indx, double *b);
extern void Erosao(int nImgIn, int nImgOut, int np, ponto *elemest);
extern void Dilatacao(int nImgIn, int nImgOut, int np, ponto *elemest);
extern void Fechamento(int nImgIn, int nImgOut, int np, ponto *strelem);
extern void DestroiObjFront(int nImgIn, int nImgOut, int np, ponto *ElemEst, int m1, int m2);
extern void CriaElemEst5Pts(double r1, double ang1, double r2, double ang2, ponto *strelem);
extern void CriaElemEst3Pts(double r, double ang, ponto *strelem);
extern void Abertura(int nImgIn, int nImgOut, int np, ponto *strelem);
extern void sort(int n, double *X);
extern void spline(float *x, float *y, int n, float yp1, float ypn, float *y2);
extern void splint(float *xa, float *ya, float *y2a, int n, float x, float *y);
extern int CalcLimiar(int nImgIn);
extern int *ivector(int nl, int nh);
extern int CriaElemEstCirc(int raio, ponto *ElemEst);
extern int CriaElemEstAro(int raio, ponto *ElemEst);
extern int InterFeixes(int nImg1, int nImg2, double *X, double *Y);
extern int CalcCompReta(int nImg, int nreta);
extern int Rotulacao(int nImgIn, int nImgOut, double minArea);
extern int TransfDist(int nImgIn, int nImgOut, int np, ponto *strelem);
extern int MedOrdpRot(int nImg1, int nImg2, int nImg3);
extern float *vector(int nl, int nh);
extern double *dvector(int nl, int nh);
extern double EstatRuido(int nImg);
extern double **dmatrix(int nrl, int nrh, int ncl, int nch);
extern double IdFreqESPRIT(int nImgIn, int nFreq, BOOL Vert);

/*****************************************************************************************************
 *****************************************************************************************************
 BIBLIOTECA DE ALGORITMOS DE VISÃO COMPUTACIONAL E AUXILIARES
```

```
 Este programa implementa um conjunto de algoritmos de visão computacional e rotinas auxiliares a partir
 dos quais se pode construir variada gama de instrumentos metrológicos  automáticos.
 ********************************************************************************************************
 ********************************************************************************************************/

#define STRICT
#include <windows.h>
#include <windowsx.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <malloc.h>
#pragma hdrstop
#include "Alg-VisaoComp.h"

/********************************************************************************************************
 ImagemBinaria
 -------------
 Verifica se a imagem é binária.
 Parâmetros:
   nImg: número da imagem
 ********************************************************************************************************/
BOOL ImagemBinaria(int nImg)
{
  BOOL bin; // TRUE se a imagem fôr binária
  int i;    // índice

  bin = TRUE;
  for (i=0; i<DimBitmap[nImg].cx*DimBitmap[nImg].cy; i++)
    if ((PixImg[nImg][i] > 0) && (PixImg[nImg][i] < 255))
      bin = FALSE;

  return(bin);
}

/********************************************************************************************************
 ApagaImagem
 -----------
 Anula os valores dos pixels de uma imagem.
 Parâmetros:
   nImg: número da imagem
 ********************************************************************************************************/
void ApagaImagem(int nImg)
{
  int sImg; // tamanho da imagem
  int i;    // índice

  sImg = DimBitmap[nImg].cx*DimBitmap[nImg].cy;

  for (i=0; i<sImg; i++)
    PixImg[nImg][i] = 0;

  return;
}

/********************************************************************************************************
 Ordena
 ------
 Coloca em ordem crescente os n elementos de um vetor X e associa aos mesmos os correspondentes valores do
 vetor agregado Y, gerando na saída o vetor ordenado Xord e seu vetor associado Yagr.
 Parâmetros:
   n    : número de elementos do vetor
   X    : vetor a ser ordenado
   Y    : vetor agregado a X
   Xord: vetor X ordenado
   Yord: vetor Y agregado
 ********************************************************************************************************/
void Ordena(int n, double *X, double *Y, double *Xord, double *Yagr)
{
  double minX;  // valor mínimo de X em uma dada iteração
  double Ya;    // correspondente valor de Y
  int i, k;     // índices
  int imin;     // índice correspondente ao valor mínimo de X

  #define infinito 1000000

  for (k=0; k<n; k++) {
    minX = infinito;
    // localiza o mínimo dos n-k últimos elementos do vetor X
    for (i=k; i<n; i++)
      if (X[i] < minX) {
        minX = X[i];
        Ya = Y[i];
```

```
        imin = i;
      }
    // permuta as posições dos elementos corrente e mínimo
    X[imin] = X[k];
    Y[imin] = Y[k];
    X[k] = minX;
    Y[k] = Ya;
    Xord[k] = X[k];
    Yagr[k] = Y[k];
  }

  #undef infinito

  return;
}

/*****************************************************************************************************************
 LimiarFixo
 ----------
 Limiariza uma imagem.
 Parâmetros:
   nImgIn : Imagem original em níveis de cinza
   nImgOut: Imagem limiarizada
   limiar : Valor do limiar
   Bin    : TRUE -> binariza a imagem
 *****************************************************************************************************************/
void LimiarFixo(int nImgIn, int nImgOut, int limiar, BOOL Bin)
{
  int i;    // índice
  int sImg; // tamanho da imagem

  sImg = DimBitmap[nImgIn].cx*DimBitmap[nImgIn].cy;

  if (Bin)  // Imagem Binária
    for (i=0; i<sImg; i++) {
      if (PixImg[nImgIn][i] > limiar)
        PixImg[nImgOut][i] = 255;
      else
        PixImg[nImgOut][i] = 0;
    }
  else      // Imagem em níveis de cinza
    for (i=0; i<sImg; i++) {
      if (PixImg[nImgIn][i] > limiar)
        PixImg[nImgOut][i] = PixImg[nImgIn][i];
      else
        PixImg[nImgOut][i] = 0;
    }

  DetCoordBordas(nImgOut);

  // Apresentação da imagem limiarizada

  SetBitmapBits(hbm[nImgOut], (DWORD)sImg, (LPSTR)PixImg[nImgOut]);
  MostraImagem(nImgOut);

  return;
}

/*****************************************************************************************************************
 DetCoordBordas
 --------------
 Determina as coordenadas das bordas da imagem segmentada nImgIn.
 Parâmetros:
   nImgIn: número da imagem
 *****************************************************************************************************************/
void DetCoordBordas(int nImgIn)
{
  int i,j,k;        // índices
  int nLin, nCol;   // número de linhas e de colunas da imagem

  nLin = DimBitmap[nImgIn].cy;
  nCol = DimBitmap[nImgIn].cx;

  for (i=0; i<nLin; i++)
    for (j=0; j<nCol; j++) {
      k = i*nCol+j;
      if (PixImg[nImgIn][k]>0) { // Objeto
        CoordBordas.XBorda[k] = (float)j;
        CoordBordas.YBorda[k] = (float)i;
      }
      else {                     // Background
        CoordBordas.XBorda[k] = -999.0F;
        CoordBordas.YBorda[k] = -999.0F;
      }
```

```
    }

  return;
}

/*****************************************************************************************************
 CalcLimiar
 ----------
 Determina, segundo o método proposto por Otsu, um valor ótimo de limiar.
 Parâmetros
   nImgIn: Imagem original em níveis de cinza
 Retorna
   limiar: Valor do limiar
 *****************************************************************************************************/
int CalcLimiar(int nImgIn)
{
  int i,j;          // índices
  long sImg;        // tamanho da imagem
  int HistG[256];   // histograma de níveis de cinza
  BYTE minG,maxG;   // valores mínimo e máximo de HistG
  int N;            // número de pixels com níveis de cinza em [minG,maxG]
  double muT;       // momento de ordem 1 do histograma completo
  double mu;        // momento de ordem 1 do histograma parcial
  double omega;     // momento de ordem zero do histograma parcial
  double sigma;     // discriminante
  double sigmaMax;  // máximo valor do discriminante
  int LimiarOtimo;  // valor ótimo do limiar

  sImg = abs(DimBitmap[nImgIn].cx*DimBitmap[nImgIn].cy);

  // Construção do histograma de níveis de cinza da imagem

  minG = 255;
  maxG = 0;
  for (i=0; i<255; i++)
    HistG[i] = 0;
  for (i=0; i<sImg; i++) {
     HistG[PixImg[nImgIn][i]]++;
     if (PixImg[nImgIn][i] <= minG)
        minG = PixImg[nImgIn][i];
     if (PixImg[nImgIn][i] >= maxG)
        maxG = PixImg[nImgIn][i];
  }

  // Determinação  do momento de primeira ordem do histograma

  muT = 0.0;
  N = 0;
  for (i=minG; i<=maxG; i++) {
    N+= HistG[i];
    muT+= (double)(i*HistG[i]);
  }
  muT/= N;

  // Determinação do valor ótimo do limiar

  sigmaMax = 0;
  for (i=minG; i<=maxG; i++) {
    omega = 0.0;
    mu = 0.0;
    for (j=minG; j<=i; j++) {
      omega+= (double)HistG[j];
      mu+= (double)(j*HistG[j]);
    }
    omega/= N;
    mu/= N;
    sigma = (muT*omega-mu)*(muT*omega-mu)/(omega*(1.0-omega));
    if (sigma > sigmaMax) {
      sigmaMax = sigma;
      LimiarOtimo= i;
    }
  }

  return LimiarOtimo;
}

/*****************************************************************************************************
 Rotulacao
 ---------
 Constrói uma imagem rotulada, partir da imagem binária original, caracterizando cada objeto por um dado
 nível de cinza. Apenas os objetos maiores que uma certa porcentagem do maior objeto da imagem são
 considerados.
 Parâmetros:
   nImgIn : imagem binária
```

```
   nImgOut: imagem rotulada
   frac: fração da área do maior objeto da imagem
 Retorna:
   número de objetos localizados
 ***************************************************************************************************************/
int Rotulacao(int nImgIn, int nImgOut, double frac)
{
  int i,j,k,m;                // índices
  int nLin, nCol;             // número de linhas e de colunas da imagem
  int sImg;                   // tamanho da imagem
  int nRot;                   // número de objetos identificados
  int p_hor;                  // pixel à esquerda do pixel considerado
  int p_ver;                  // pixel acima do pixel considerado
  int p_dia;                  // pixel acima e à esquerda do pixel considerado
  int p1, p2;                 // valores de pixel
  HGLOBAL hM1,hM2,hM3;        // manipuladores de memória
  int *Rotulo;                // "imagem" de rótulos
  int *AreaObj;               // áreas dos objetos detectados
  int *lRot;                  // vetor associando rótulos novos e antigos
  int AreaMin,AreaMax;        // intervalos de áreas de objeto de interesse
  BOOL novo;                  // TRUE quando se localiza um novo objeto
  int gmin;                   // nível de cinza mínimo
  LPBYTE imgA;                // imagem final apresentada

  // Alocação de memória para os rótulos

  nLin = DimBitmap[nImgIn].cy;
  nCol = DimBitmap[nImgIn].cx;
  sImg = nLin*nCol;
  hM1 = GlobalAlloc(GHND, (DWORD)(sImg*sizeof(int)));
  Rotulo = (int *) GlobalLock(hM1);

  // Identificação de Objetos Conectados

  nRot = 1;
  for (i=1; i<nLin; i++)
    for (j=1; j<nCol; j++) {
      k = i*nCol+j;
      p_hor = Rotulo[k-1];
      p_ver = Rotulo[k-nCol];
      p_dia = Rotulo[k-nCol-1];
      if (PixImg[nImgIn][k]==255) {
        if (p_dia != 0)
          // Pixel conectado diagonalmente a um objeto já rotulado
          Rotulo[k] = p_dia;
        else if ((p_hor==0) && (p_ver==0)) {
          // Pixel de um novo objeto !
          Rotulo[k] = nRot;
          nRot++;
        }
        else if ((p_hor!=0) && (p_ver==0))
          // Pixel conectado à esquerda com objeto já rotulado
          Rotulo[k] = p_hor;
        else if ((p_hor==0) && (p_ver!=0))
          // Pixel conectado acima com objeto já rotulado
          Rotulo[k] = p_ver;
        else if ((p_hor!=0) && (p_ver!=0)) {
          // Pixel conectado à esquerda e acima com objeto já rotulado
          Rotulo[k] = p_ver;
          if (p_hor != p_ver) {
            // Existem dois rótulos equivalentes: p_hor e p_ver
            p1 = __min(p_hor, p_ver);
            p2 = __max(p_hor, p_ver);
            for (m=0; m<=k; m++)
              // Equiparação dos pixels do mesmo objeto com rótulos distintos
              if (Rotulo[m] == p2)
                Rotulo[m] = p1;
          }
        }
      }
    }

  // Determinação das áreas de cada objeto e da área do maior objeto

  hM2 = GlobalAlloc(GHND, (DWORD)((nRot+1)*sizeof(int)));
  AreaObj = (int *) GlobalLock(hM2);
  for (i=0; i<sImg; i++)
    if (Rotulo[i] != 0)
      AreaObj[Rotulo[i]]++;
  AreaMax = 0;
  for (i=1; i<nRot; i++)
    if (AreaObj[i]>AreaMax)
      AreaMax = AreaObj[i];
  AreaMin = (int)(frac*AreaMax);
```

```
  GlobalUnlock(hM2);

  // Eliminação dos objetos com área menor que o mínimo admissível

  for (i=0; i<sImg; i++)
    if ((Rotulo[i] != 0) && (AreaObj[Rotulo[i]] < AreaMin))
      Rotulo[i] = 0;

  // Criação de uma Lista de Rótulos

  hM3 = GlobalAlloc(GHND, (DWORD)((nRot+1)*sizeof(int)));
  lRot = (int *) GlobalLock(hM3);
  nRot = 0;
  for (i=0; i<sImg; i++)
    if (Rotulo[i] != 0) {
      novo = TRUE;
      for (j=0; j<nRot; j++)
        if (Rotulo[i] == lRot[j])
          novo = FALSE;
      if (novo) {
        lRot[j] = Rotulo[i];
        nRot++;
      }
    }

  // Construção da Imagem de Objetos

  if (nRot < 255)
    gmin = 255-nRot;
  else
    MessageBox(NULL, "Número de objetos > 256", NULL, MB_OK);
  for (i=0; i<sImg; i++)
    if (Rotulo[i] != 0) {
      j=0;
      while (Rotulo[i] != lRot[j])
        j++;
      PixImg[nImgOut][i] = (BYTE)(gmin+j);
    }

  // Apresentação da imagem de objetos rotulados

  hM2 = GlobalAlloc(GHND, (DWORD)sImg);
  imgA = (LPBYTE) GlobalLock(hM2);
  for (i=0; i<sImg; i++)
    imgA[i] = ConvIndPal[PixImg[nImgOut][i]];
  SetBitmapBits(hbm[nImgOut], (DWORD)sImg, (LPSTR)imgA);
  MostraImagem(nImgOut);

  // Liberação de memória

  GlobalUnlock(hM1);
  GlobalUnlock(hM2);
  GlobalUnlock(hM3);

  return nRot;
}

/*****************************************************************************************************
 FiltroMedia
 -----------
 Aplica filtro-média a uma imagem.
 Parâmetros:
   nImgIn : imagem original
   nImgOut: imagem filtrada
   w      : largura da janela
 *****************************************************************************************************/
void FiltroMedia(int nImgIn, int nImgOut, int w)
{
  int i,j,k,m,n;          // índices
  int nCol, nLin;         // numero de colunas e de linhas das imagens
  int sImg;               // tamanho da imagem
  int mw;                 // meia-largura da janela
  double w2;              // área da janela
  double soma;            // valor local da convolução
  HGLOBAL hM;             // manipulador de memória
  LPBYTE imgA;            // imagem final apresentada

  // Inicialização da imagem filtrada nImgOut

  nCol = DimBitmap[nImgIn].cx;
  nLin = DimBitmap[nImgIn].cy;
  sImg = nCol*nLin;
  for (i=0; i<sImg; i++)
    PixImg[nImgOut][i] = PixImg[nImgIn][i];
```

```
  // Convolução da imagem com o filtro média de meia largura mw

  mw = w/2;
  w2 = w*w;
  for (i=mw; i<nLin-mw; i++)
    for (j=mw; j<nCol-mw; j++) {
      soma = 0.0;
      for (m=-mw; m<=mw; m++)
        for (n=-mw; n<=mw; n++) {
          k = (i+m)*nCol+j+n;
          soma+= PixImg[nImgIn][k];
        }
      k = i*nCol+j;
      n = (BYTE)(soma/w2);
      PixImg[nImgOut][k] =(BYTE)n;
    }

  // Apresentação da imagem filtrada

  hM = GlobalAlloc(GHND, (DWORD)sImg);
  imgA = (LPBYTE) GlobalLock(hM);
  for (i=0; i<sImg; i++)
    imgA[i] = ConvIndPal[PixImg[nImgOut][i]];
  SetBitmapBits(hbm[nImgOut], (DWORD)sImg, (LPSTR)imgA);
  MostraImagem(nImgOut);

  // Liberação de memória

  GlobalUnlock(hM);

  return;
}

/*****************************************************************************************************************
 FiltroMediana
 -------------                                                                                                    "
 Aplica filtro-mediana a uma imagem.
 Parâmetros:
   nImgIn : imagem original
   nImgOut: imagem filtrada
   w      : largura da janela
*****************************************************************************************************************/
void FiltroMediana(int nImgIn, int nImgOut, int w)
{
  int i,j,k,l,m,n,p;     // índices
  int nCol, nLin;        // numero de colunas e de linhas das imagens
  int sImg;              // tamanho da imagem
  int mw;                // meia-largura da janela de convolução
  HGLOBAL hM1,hM2,hM3;   // manipuladores de memória
  double *V;             // pixels de uma dada vizinhança
  double *Vord;          // pixels ordenados de uma dada vizinhança
  int w2;                // posição média dos vetores V e Vord
  LPBYTE imgA;           // imagem final apresentada

  // Inicialização da imagem filtrada e dos vetores V e Vord

  nCol = DimBitmap[nImgIn].cx;
  nLin = DimBitmap[nImgIn].cy;
  sImg = nLin*nCol;
  for (i=0; i<sImg; i++)
    PixImg[nImgOut][i] = PixImg[nImgIn][i];
  hM1 = GlobalAlloc(GHND, (DWORD)(w*w*sizeof(double)));
  V = (double *) GlobalLock(hM1);
  hM2 = GlobalAlloc(GHND, (DWORD)(w*w*sizeof(double)));
  Vord = (double *) GlobalLock(hM2);

  // Convolução da imagem com um filtro mediana de lagura w

  mw = w/2;
  w2 = (w*w)/2;
  for (i=mw; i<nLin-mw; i++)
    for (j=mw; j<nCol-mw; j++) {
      k = i*nCol+j;
      p = 0;
      for (m=-mw; m<=mw; m++)
        for (n=-mw; n<=mw; n++) {
          l = (i+m)*nCol+(j+n);
          V[p]= (double)(PixImg[nImgIn][l]);
          p++;
        }
      Ordena(p, V, V, Vord, Vord);
      PixImg[nImgOut][k] = (BYTE)Vord[w2];
    }
```

```
  // Apresentação da imagem filtrada

  hM3 = GlobalAlloc(GHND, (DWORD)sImg);
  imgA = (LPBYTE) GlobalLock(hM3);
  for (i=0; i<sImg; i++)
    imgA[i] = ConvIndPal[PixImg[nImgOut][i]];
  SetBitmapBits(hbm[nImgOut], (DWORD)sImg, (LPSTR)imgA);
  MostraImagem(nImgOut);

  // Liberação de  memória

  GlobalUnlock(hM1);
  GlobalUnlock(hM2);
  GlobalUnlock(hM3);

  return;
}

/*****************************************************************************************************************
 Op_BOOL
 -------
 Combina duas imagens por meio de operações booleanas.
 Parâmetros:
   numOp: tipo de operação (OR,AND,XOR,NOT)
   nImg1: imagem original
   nImg2: imagem original
   nImg3: imagem resultante
 *****************************************************************************************************************/
void Op_BOOL(int numOp, int nImg1, int nImg2, int nImgOut)
{
  int k;              // índice
  int sImg;           // tamanho da imagem
  HGLOBAL hM;         // manipulador de memória
  LPBYTE imgA;        // imagem final apresentada

  // Inicialização da imagem resultante

  sImg = DimBitmap[nImg1].cy*DimBitmap[nImg1].cx;
  for (k=0; k<sImg; k++)
    PixImg[nImgOut][k] = 0;
  hM = GlobalAlloc(GHND, (DWORD)sImg);
  imgA = (LPBYTE) GlobalLock(hM);

  // Aplicação da operação booleana

  switch (numOp) {
    case 0: // OR
      for (k=0; k<sImg; k++) {
        PixImg[nImgOut][k] = __max(PixImg[nImg1][k],PixImg[nImg2][k]);
        imgA[k] = ConvIndPal[PixImg[nImgOut][k]];
      }
      break;
    case 1: // AND
      for (k=0; k<sImg; k++) {
        PixImg[nImgOut][k] = __min(PixImg[nImg1][k],PixImg[nImg2][k]);
        imgA[k] = ConvIndPal[PixImg[nImgOut][k]];
      }
      break;
    case 2: // XOR
      for(k=0; k<sImg; k++) {
        PixImg[nImgOut][k] = (BYTE)abs(PixImg[nImg1][k]-PixImg[nImg2][k]);
        imgA[k] = ConvIndPal[PixImg[nImgOut][k]];
      }
      break;
    case 3: // NOT
      for (k=0; k<sImg; k++) {
        PixImg[nImgOut][k] = (BYTE)abs(PixImg[nImg1][k]-255);
        imgA[k] = ConvIndPal[PixImg[nImgOut][k]];
      }
      break;
  }

  // Apresentação da imagem resultante

  SetBitmapBits(hbm[nImgOut], (DWORD)sImg, (LPSTR)imgA);
  MostraImagem(nImgOut);

  // Liberação de memória

  GlobalUnlock(hM);

  return;
}
```

```
/*****************************************************************************************************
 ImagemdeRetas
 -------------
 Gera imagem com o feixe de retas localizadas na imagem original.
 Parâmetros:
   nImgIn : imagem original
   nImgOut: imagem com as retas localizadas em nImgIn
   nRetas : número de retas
   Theta  : coordenadas angulares das retas
   Rho    : coordenadas radiais das retas
 *****************************************************************************************************/
void ImagemdeRetas(int nImgIn, int nImgOut, int nRetas, double *Theta, double *Rho)
{
  int i,j,k;            // índices
  int nLin, nCol;       // número de linhas e colunas da imagem
  double x0;            // cruzamento da reta com o eixo x
  double y0;            // cruzamento da reta com o eixo y
  double x,y;           // coordenadas de um ponto da reta
  int ix,iy;            // valores inteiros de x,y

  #define EPSILON 0.1   // tolerância angular

  // Alocação de memória e inicialização da imagem de retas

  nLin = DimBitmap[nImgIn].cy;
  nCol = DimBitmap[nImgIn].cx;
  DimBitmap[nImgOut].cx = nCol;
  DimBitmap[nImgOut].cy = nLin;
  hbm[nImgOut] = CreateBitmap(nCol, nLin, 1, 8,(VOID *)NULL);
  for (j=0; j<nLin*nCol; j++)
    PixImg[nImgOut][j] = 0;

  // Geração das retas normais a cada um dos vetores (Rho(i),Theta)

  for (k=0; k<nRetas; k++) {
    i = (Rho[k]-floor(Rho[k]) < 0.5) ? (int)Rho[k] : (int)Rho[k]+1;
    if (fabs(Theta[k]) <= EPSILON)                    // reta vertical
      for (j=0; j<nLin; j++)
        PixImg[nImgOut][j*nCol+i] = 255;
    else if (fabs(fabs(Theta[k])-90) <= EPSILON)      // reta horizontal
      for (j=0; j<nCol; j++)
        PixImg[nImgOut][i*nCol+j] = 255;
    else {                                            // reta inclinada
      x0 = Rho[k]/cos(Theta[k]*GRAD);
      y0 = Rho[k]/sin(Theta[k]*GRAD);
      if ((fabs(Theta[k]) >= 45.00) && (fabs(Theta[k]) <=135.00))
        for (j=0; j<nCol; j++) {
          y = y0*(1.0-(double)j/x0);
          iy = (y-floor(y) < 0.5) ? (int)y : (int)y+1;
          if ((iy>0) && (iy<nLin))
            PixImg[nImgOut][iy*nCol+j] = 255;
        }
      else
        for (i=0; i<nLin; i++) {
          x = x0*(1.0-(double)i/y0);
          ix = (x-floor(x) < 0.5) ? (int)x : (int)x+1;
          if ((ix>0) && (ix<nCol))
            PixImg[nImgOut][i*nCol+(int)x] = 255;
        }
    }
  }

  // Apresentação da imagem de retas

  SetBitmapBits(hbm[nImgOut], (DWORD)(nCol*nLin), (LPSTR)PixImg[nImgOut]);
  IndImgSel = nImgOut;
  MostraImagem(nImgOut);

  #undef EPSILON

  return;

}

/*****************************************************************************************************
 FeixeMedio
 ----------
 Gera a imagem de um feixe de retas na posição média entre os feixes de retas de duas imagens dadas.
 Parâmetros:
   nImg1 : imagem do primeiro feixe
   nImg2 : imagem do segundo feixe
   nImgOut: imagem do "feixe médio"
 *****************************************************************************************************/
```

```
void FeixeMedio(int nImg1, int nImg2, int nImgOut)
{
  int n;            // número de retas dos feixes
  int i;            // índice
  double sig1,sig2; // variâncias de cada reta ajustada nas imagens 1 e 2
  double sig;       // variância da reta medial

  // Caracterização geométrica do "feixe de retas médio"

  n = FeixeRetas[nImg1].nRetasImg;
  FeixeRetas[nImgOut].nRetasImg = n;
  for (i=0; i<n; i++)   {
    FeixeRetas[nImgOut].Raio  [i] = 0.5*(FeixeRetas[nImg1].Raio  [i] + FeixeRetas[nImg2].Raio  [i]);
    FeixeRetas[nImgOut].Angulo[i] = 0.5*(FeixeRetas[nImg1].Angulo[i] + FeixeRetas[nImg2].Angulo[i]);
    sig1 = FeixeRetas[nImg1].Sigma[i]*FeixeRetas[nImg1].Sigma[i];
    sig2 = FeixeRetas[nImg2].Sigma[i]*FeixeRetas[nImg2].Sigma[i];
    sig  = (sig1*FeixeRetas[nImg1].nPts[i]+sig2*FeixeRetas[nImg2].nPts[i])/
           (double)(FeixeRetas[nImg1].nPts[i]+FeixeRetas[nImg1].nPts[i]);
    FeixeRetas[nImgOut].Sigma [i] = sqrt(sig);;
  }

  // Apresentação da imagem correspondente ao "feixe de retas mediais"

  sprintf(NomeImg[nImgOut], "Img%d = Feixe de Retas Mediais", nImgOut);
  ImagemdeRetas(nImg1, nImgOut, FeixeRetas[nImgOut].nRetasImg,
                FeixeRetas[nImgOut].Angulo, FeixeRetas[nImgOut].Raio);

  return;
}

/*****************************************************************************************************
 InterFeixes
 -----------
 Obtém as intersecções de dois feixes de retas.
 Parâmetros:
   nImg1 : imagem do feixe 1
   nImg2 : imagem do feixe 2
   X,Y   : coordenadas dos pontos de intersecção
 *****************************************************************************************************/
int InterFeixes(int nImg1, int nImg2, double *X, double *Y)
{
  int n1, n2;                    // número de retas dos feixes 1 e 2
  double xp, yp;                 // coordenadas dos pontos de intersecção de duas retas
  int i,j,k;                     // índices
  double Ang1[NMAX_RETAS_IMG];   // ângulos das retas do feixe 1
  double Raio1[NMAX_RETAS_IMG];  // distâncias radiais à origem das retas do feixe 1
  double Ang2[NMAX_RETAS_IMG];   // ângulos das retas do feixe 2
  double Raio2[NMAX_RETAS_IMG];  // distâncias radiais à origem das retas do feixe 2

  // Atribuição de valores a variáveis auxiliares

  n1 = FeixeRetas[nImg1].nRetasImg;
  for (i=0; i<n1; i++) {
    Ang1[i]  = FeixeRetas[nImg1].Angulo[i];
    Raio1[i] = FeixeRetas[nImg1].Raio[i];
  }
  n2 = FeixeRetas[nImg2].nRetasImg;
  for (i=0; i<n2; i++) {
    Ang2[i]  = FeixeRetas[nImg2].Angulo[i];
    Raio2[i] = FeixeRetas[nImg2].Raio[i];
  }

  // Determinação das intersecções das retas dos feixes 1 e 2

  k = 0;
  for (i=0; i<n1; i++)
    for (j=0; j<n2; j++)
      if (InterRetas(nImg1, Ang1[i], Raio1[i], Ang2[j], Raio2[j], &xp, &yp)) {
        X[k] = xp;
        Y[k] = yp;
        k++;
      }

  return k;
}

/*****************************************************************************************************
 InterRetas
 ----------
 Obtém o ponto de intersecção de duas retas, verificando se  é interior às fronteiras da imagem-referência.
 Parâmetros:
   nImg1 : imagem referência
   Ang1  : orientação da reta 1
   Raio1 : distância da reta 1 à origem
```

```
   Ang2  : orientação da reta 2
   Raio2 : distância da reta 2 à origem
   xp,yp : coordenadas da intersecção das retas
 ***************************************************************************************************/
BOOL InterRetas(int nImg1, double Ang1, double R1, double Ang2, double R2, double *x, double *y)
{
  int tipo1, tipo2;          // tipos de equação de reta para as retas 1 e 2
  double a1, b1, a2, b2;     // parâmetros das equações das retas

  // Classifica as retas em 4 tipos:
  // 1. x = a       : vertical
  // 2. y = a       : horizontal
  // 3. x = a + b.y : inclinada
  // 4. y = a + b.x : inclinada

  tipo1 = ClassificaRetas(Ang1, R1, &a1, &b1);
  tipo2 = ClassificaRetas(Ang2, R2, &a2, &b2);

  // Determinação das coordenadas do ponto de intersecção

  if ((tipo1==1) && (tipo2==1))          // reta 1: vertical ; reta 2: vertical
    return FALSE;
  else if ((tipo1==1) && (tipo2==2)) {   // reta 1: vertical ; reta 2: horizontal
    *x = a1;
    *y = a2;
  }
  else if ((tipo1==1) && (tipo2==3)) {   // reta 1: vertical ; reta 2: x = a2 + b2.y
    *x = a1;
    *y = (a1-a2)/b2;
  }
  else if ((tipo1==1) && (tipo2==4)) {   // reta 1: vertical ; reta 2: y = a2 + b2.x
    *x = a1;
    *y = a2+b2*a1;
  }
  else if ((tipo1==2) && (tipo2==1)) {   // reta 1: horizontal ; reta 2: vertical
    *x = a2;
    *y = a1;
  }
  else if ((tipo1==2) && (tipo2==2))     // reta 1: horizontal ; reta 2: horizontal
    return FALSE;
  else if ((tipo1==2) && (tipo2==3)) {   // reta 1: horizontal ; reta 2: x = a2 + b2.y
    *y = a1;
    *x = a2+b2*a1;
  }
  else if ((tipo1==2) && (tipo2==4)) {   // reta 1: horizontal ; reta 2: y = a2 + b2.x
    *y = a1;
    *x = (a1-a2)/b2;
  }
  else if ((tipo1==3) && (tipo2==1)) {   // reta 1: x = a1 + b1.y ; reta 2: vertical
    *x = a2;
    *y = (a2-a1)/b1;
  }
  else if ((tipo1==3) && (tipo2==2)) {   // reta 1: x = a1 + b1.y ; reta 2: horizontal
    *y = a2;
    *x = a1+b1*a2;
  }
  else if ((tipo1==3) && (tipo2==3)) {   // reta 1: x = a1 + b1.y ; reta 2: x = a2 + b2.y
    *x = (a2*b1-a1*b2)/(b1-b2);
    *y = (a2-a1)/(b1-b2);
  }
  else if ((tipo1==3) && (tipo2==4)) {   // reta 1: x = a1 + b1.y ; reta 2: y = a2 + b2.x
    *x = (a1+b1*a2)/(1-b1*b2);
    *y = (a2+a1*b2)/(1-b1*b2);
  }
  else if ((tipo1==4) && (tipo2==1)) {   // reta 1: y = a1 + b1.x ; reta 2: vertical
    *x = a2;
    *y = a1+b1*a2;
  }
  else if ((tipo1==4) && (tipo2==2)) {   // reta 1: y = a1 + b1.x ; reta 2: horizontal
    *y = a2;
    *x = (a2-a1)/b1;
  }
  else if ((tipo1==4) && (tipo2==3)) {   // reta 1: y = a1 + b1.x ; reta 2: x = a2 + b2.y
    *x = (a2+a1*b2)/(1-b1*b2);
    *y = (a1+b1*a2)/(1-b1*b2);
  }
  else if ((tipo1==4) && (tipo2==4)) {   // reta 1: y = a1 + b1.x ; reta 2: y = a2 + b2.x
    *x = (a2-a1)/(b1-b2);
    *y = (b1*a2-b2*a1)/(b1-b2);
  }

  // Verifica se o ponto é interior ao quadro definidor da imagem

  if ((*x>=0) && (*x<=DimBitmap[nImg1].cx) && (*y>=0) && (*y<=DimBitmap[nImg1].cy))
```

```
    return TRUE;
  else
    return FALSE;
}

/*****************************************************************************************************************
 ClassificaRetas
 ---------------
 Classifica a equação da reta em quatro categorias:
 1) x = a       : vertical
 2) y = a       : horizontal
 3) x = a + b.y : inclinada
 4) y = a + b.x : inclinada
 Parâmetros:
   Angulo:  orientação da reta
   Raio   : distância da reta à origem
   a, b   : parâmetros da equação da reta
 *****************************************************************************************************************/
int ClassificaRetas(double Angulo, double Raio, double *a, double *b)
{
  int tipo;             // 1)vertical; 2)horizontal; 3)inclinada

  #define EPSILON 0.1   // tolerância angular

  if (fabs(Angulo) <= EPSILON) {
    // reta vertical: x = Raio = a
    *a = Raio;
    *b = 0;
    tipo = 1;
  }
  else if (fabs(fabs(Angulo)-90) <= EPSILON) {
    // reta horizontal: y = Raio = a
    *a = Raio;
    *b = 0;
    tipo = 2;
  }
  if ((fabs(Angulo) <= 45) || (fabs(Angulo) >= 135)) {
    // reta inclinada: x = a + by
    *a = Raio/cos(Angulo*GRAD);
    *b = -tan(Angulo*GRAD);
    tipo = 3;
  }
  else {
    // reta inclinada: y = a + bx
    *a = Raio/sin(Angulo*GRAD);
    *b = -1.0/tan(Angulo*GRAD);
    tipo = 4;
  }

  #undef EPSILON

  return tipo;
}

/*****************************************************************************************************************
 CalcCompReta
 ------------
 Calcula o comprimento de uma reta de uma imagem digital.
 Parâmetros:
   nImg : número da imagem
   nreta: índice da reta na imagem nImg
 Retorna:
   perímetro da reta digital
 *****************************************************************************************************************/
int CalcCompReta(int nImg, int nreta)
{
  int nLin,nCol;     // número de linhas e de colunas da imagem
  double r,ang;      // coordenadas polares da reta
  double x0,y0;      // intersecções da reta com os eixos x e y
  int m;             // número de pontos da reta digital
  int i,j;           // índices
  int ix,iy;         // coordenadas inteiras de um ponto da reta

  nLin = DimBitmap[nImg].cy;
  nCol = DimBitmap[nImg].cx;
  r = FeixeRetas[nImg].Raio[nreta];
  ang = FeixeRetas[nImg].Angulo[nreta];
  x0 = (fabs(ang)!=90.0) ? r/cos(ang*GRAD) : 1.0e100;
  y0 = ((ang!=0.0) || (fabs(ang)!=270.0)) ? r/sin(ang*GRAD) : 1.0e100;
  m = 0;
  if ((fabs(ang) >= 45.00) && (fabs(ang) <=135.00))
    for (j=0; j<nCol; j++) {
      iy = (int)(y0*(1.0-(double)j/x0));
      if ((iy>=0) && (iy<nLin))
```

```
        m++;
      }
    else
      for (i=0; i<nLin; i++) {
        ix = (int)(x0*(1.0-(double)i/y0));
        if ((ix>=0) && (ix<nCol))
          m++;
      }

    return m;
}

/*****************************************************************************************************
 AssociaImagemObjeto
 -------------------
 Estabelece correspondência entre os dados de calibração da microescala-referência e as retas da imagem.
 Parâmetros:
   nImg   : imagem de feixe de retas
   fLinear: tipo de feixe  -> TRUE: linear; FALSE: angular
   medReal: tipo de medida -> TRUE: real   ; FALSE: NOMINAL
 *****************************************************************************************************/
void AssociaImagemObjeto(int nImg, BOOL fLinear, BOOL medReal)
{
  int n;                          // número de retas da imagem
  double Ang [NMAX_RETAS_IMG];    // angulos das retas do feixe original
  double Rad [NMAX_RETAS_IMG];    // raios das retas do feixe original
  double Ang1[NMAX_RETAS_IMG];    // ângulos das retas do feixe ordenado
  double Rad1[NMAX_RETAS_IMG];    // raios das retas do feixe ordenado
  int i,j,k;                      // índices
  int nTracos;                    // número de tracos intermediários não calibrados
  double fracao;                  // fracao de tracos não calibrados

  // Váriáveis auxiliares

  n = FeixeRetas[nImg].nRetasImg;
  for (i=0; i<n; i++) {
    Ang[i] = FeixeRetas[nImg].Angulo[i];
    Rad[i] = FeixeRetas[nImg].Raio[i];
  }

  // Ordenacao das retas do feixe da imagem nImg

  if (fLinear) {
    Ordena(n, Rad, Ang, Rad1, Ang1);     // Ordena segundo raios crescentes
    FeixeRetas[nImg].tipo = 1;
  }
  else {
    Ordena(n, Ang, Rad, Ang1, Rad1);     // Ordena segundo ângulos crescentes
    FeixeRetas[nImg].tipo = 0;
  }
  for (i=0; i<n; i++) {
    FeixeRetas[nImg].Angulo[i] = Ang1[i];
    FeixeRetas[nImg].Raio[i]   = Rad1[i];
  }

  // Atribuição dos valores medidos da escala às retas correspondentes da imagem

  FeixeRetas[nImg].Medida[0] = (medReal) ? Escala.real[0] : Escala.nominal[0];

  i=1;
  j=1;
  while (i<n) {
    fracao = DistTracos/(Escala.nominal[j]-Escala.nominal[j-1]);
    nTracos = (int)((Escala.nominal[j]-Escala.nominal[j-1])/DistTracos-0.99);
    if ((fracao<1.01) && (fracao>=0.99))     {
      FeixeRetas[nImg].Medida[i] = (medReal) ? Escala.real[j] : Escala.nominal[j];
      i++;  // nova reta
    }
    else {
      for (k=1; k<=nTracos+1; k++) {
        FeixeRetas[nImg].Medida[i] = (medReal) ?
          Escala.real   [j-1]+(Escala.real   [j]-Escala.real   [j-1])*fracao*k:
          Escala.nominal[j-1]+(Escala.nominal[j]-Escala.nominal[j-1])*fracao*k;
        i++; // nova reta
      }
    }
    j++;    // novo traço da escala
  }
}

/*****************************************************************************************************
 EstatRuido
 ----------
 Determina o desvio-padrão do ruído Gaussiano de uma imagem.
```

```
 Parâmetros:
   nImg: imagem do ruído
 ***************************************************************************************************************/
double EstatRuido(int nImg)
{
  int i,j,k;         // índices
  int nLin, nCol;    // número de linhas e de colunas da imagem
  double sigma;      // desvio-padrão do ruído

  nLin = DimBitmap[nImg].cy;
  nCol = DimBitmap[nImg].cx;

  sigma = 0;
  for (i=0; i<nLin; i++)
    for (j=0; j<nCol; j++) {
      k = i*nCol+j;
      sigma+= PixImg[nImg][k]*PixImg[nImg][k];
    }

  sigma = sqrt(sigma/(2.0*nLin*nCol));

  return sigma;
}

/***************************************************************************************************************
 Reconstrucao
 ------------
 Reconstrói a imagem digitalizada utilizando um filtro polinomial de quarto grau proposto por Nalwa et
 Binford.
 Parâmetros:
   nImgIn : número da imagem original
   nImgOut: número da imagem apresentada
 Variáveis Globais:
   imgRec
 ***************************************************************************************************************/
void Reconstrucao(int nImgIn, int nImgOut)
{
  int nCol, nLin;        // número de linhas e de colunas da imagem original
  int sImg;              // tamanho da imagem original
  int nCol2, nLin2;      // número de linhas e de colunas da imagem reconstruída
  int i,j,k,I,J,K,a,b;   // índices
  double cnz;            // níveis de cinza da imagem reconstruída
  HGLOBAL hM1;           // manipulador de memória
  LPBYTE imgAltaRes;     // imagem original em alta resolução
  LPBYTE imgA;           // imagem final apresentada

  // Alocação de memória para as imagens

  nCol = DimBitmap[nImgIn].cx;
  nLin = DimBitmap[nImgIn].cy;
  sImg = nLin*nCol;
  nCol2 = 2*nCol;
  nLin2 = 2*nLin;
  hM1 = GlobalAlloc(GHND, (DWORD)(4*sImg));
  imgAltaRes = (LPBYTE) GlobalLock(hM1);

  // Cópia dos pixels da imagem original para os pixels correspondentes da imagem de alta resolução

  for (i=0; i<nLin; i++) {
    I = 2*i;
    for (j=0; j<nCol; j++) {
      J = 2*j;
      K = I*nCol2+J;
      k = i*nCol+j;
      imgAltaRes[K] = PixImg[nImgIn][k];
      imgRec[K] = imgAltaRes[K];
    }
  }

  // Interpolação dos pixels da imagem original usando a máscara de reconstrução

  for (i=3; i<nLin2-3; i++)
    for (j=3; j<nCol2-3; j++) {
      cnz =              imgAltaRes[i*nCol2+j] +
            0.00390625*(imgAltaRes[(i-3)*nCol2+j-3] + imgAltaRes[(i-3)*nCol2+j+3] +
                        imgAltaRes[(i+3)*nCol2+j-3] + imgAltaRes[(i+3)*nCol2+j+3])-
            0.03515625*(imgAltaRes[(i-3)*nCol2+j-1] + imgAltaRes[(i-3)*nCol2+j+1] +
                        imgAltaRes[(i+3)*nCol2+j-1] + imgAltaRes[(i+3)*nCol2+j+1] +
                        imgAltaRes[(i-1)*nCol2+j-3] + imgAltaRes[(i-1)*nCol2+j+3] +
                        imgAltaRes[(i+1)*nCol2+j-3] + imgAltaRes[(i+1)*nCol2+j+3])-
            0.06250000*(imgAltaRes[(i-3)*nCol2+j]   + imgAltaRes[(i+3)*nCol2+j] +
                        imgAltaRes[i*nCol2+j-3]     + imgAltaRes[i*nCol2+j+3]) +
            0.56250000*(imgAltaRes[(i-1)*nCol2+j]   + imgAltaRes[(i+1)*nCol2+j] +
                        imgAltaRes[i*nCol2+j-1]     + imgAltaRes[i*nCol2+j+1])  +
```

```
            0.31640625*(imgAltaRes[(i-1)*nCol2+j-1] + imgAltaRes[(i-1)*nCol2+j+1] +
                        imgAltaRes[(i+1)*nCol2+j-1] + imgAltaRes[(i+1)*nCol2+j+1]);
      imgRec[i*nCol2+j] = (BYTE)cnz;
    }
  for (i=0; i<3; i++)
    for (j=0; j<nCol2; j++)
      imgRec[i*nCol2+j] = imgRec[3*nCol2+j];
  for (i=nLin2-3; i<nLin2; i++)
    for (j=0; j<nCol2; j++)
      imgRec[i*nCol2+j] = imgRec[(nLin2-3)*nCol2+j];


  GlobalUnlock(hM1);

  // Apresentação, em 4 etapas, da imagem reconstruída

  hM1 = GlobalAlloc(GHND, (DWORD)sImg);
  imgA = (LPBYTE) GlobalLock(hM1);
  for (a=0; a<=nLin; a+=nLin)
    for (b=0; b<=nCol; b+=nCol) {
      for (i=0; i<nLin; i++)
        for (j=0; j<nCol; j++) {
          PixImg[nImgOut][i*nCol+j] = imgRec[(i+a)*nCol2+j+b];
          imgA[i*nCol+j] = ConvIndPal[imgRec[(i+a)*nCol2+j+b]];
        }
      SetBitmapBits(hbm[nImgOut], (DWORD)sImg, (LPSTR)imgA);
      sprintf(NomeImg[nImgOut], "Imagem reconstruída: referência (linha %d, coluna %d)", a, b);
      MostraImagem(nImgOut);
      Sleep(2000);
    }

  // Liberação de memória

  GlobalUnlock(hM1);

  return;
}

/*****************************************************************************************************************
 DBordasRoberts
 --------------
 Detecta as bordas da imagem com os operadores de Roberts.
 Parâmetros:
   ImgIn     : número da imagem original
   ImgOut    : número da imagem de bordas
   tipBorda  : 0)borda escuro-claro; 1)borda claro-escuro; 2)dupla borda
   Bin       : TRUE-> binariza a imagem de bordas
   pcFBordas: porcentagem tolerada de falsas bordas
   Rec       : TRUE-> a imagem ImgIn foi reconstruída
 Variáveis Globais:
   imgRec, CoordBordas
 *****************************************************************************************************************/
void DBordasRoberts(int nImgIn, int nImgOut, int tipBorda, BOOL Bin, double pcFBordas, BOOL Rec)
{
  int i,j,k;                                  // índices
  HGLOBAL hM1,hM2,hM3;                        // manipuladores de memória
  int nLin, nCol;                             // número de linhas e colunas da imagem original
  int sImg;                                   // tamanho da imagem
  double Gx, Gy;                              // gradientes da imagem segundo x e y
  float *G;                                   // matriz com os valores de magnitude do gradiente da imagem
  float Gmax;                                 // máximo valor de G
  unsigned char *D;                           // direção do vetor Gradiente: 0-> 0; 1-> 45 ... 7-> 315
  int limiar;                                 // limiar de separação de bordas-ruído
  double ang;                                 // ângulo do vetor gradiente relativamente ao eixo x
  float coef;                                 // coeficiente aplicado às coordenadas dos pixels da imagem
  LPBYTE imgBFinas;                           // imagem de bordas afinadas
  float (*pfGrad)(double x,double y);         // ponteiro para função que calcula gradiente

  // Cálculo dos parâmetros básicos utilizados pelo algoritmo de detecção de bordas

  sImg = CalcParImgBordas(nImgIn,Rec, &nCol, &nLin, &coef);

  // Seleção do método de cálculo do gradiente em função do tipo de borda

  switch(tipBorda) {
    case 0: pfGrad = CalcMagGrad1; break;
    case 1: pfGrad = CalcMagGrad2; break;
    case 2: pfGrad = CalcMagGrad3; break;
  }

  // Alocação de memória para as matrizes gradiente G e direção D

  hM1 = GlobalAlloc(GHND, (DWORD)(sImg*sizeof(float)));
  G = (float*) GlobalLock(hM1);
```

```
  hM2 = GlobalAlloc(GHND, (DWORD)(sImg));
  D = (unsigned char *) GlobalLock(hM2);

  // Cálculo da matriz Gradiente G, da matriz direção D e das coordenadas das bordas

  for (i=1; i<nLin-1; i++)
    for (j=1; j<nCol-1; j++) {
      k = i*nCol+j;
      Gx = imgRec[k+1]    - imgRec[k];
      Gy = imgRec[k+nCol] - imgRec[k];
      G[k] = pfGrad(Gx, Gy);
      if (G[k] > 0) {
        ang = atan(fabs(Gy/Gx))/GRAD;
        D[k] = CalcOriGrad(Gx, Gy, ang);
        CoordBordas.XBorda[k] = coef*(float)j;
        CoordBordas.YBorda[k] = coef*(float)i;
      }
      else {
        CoordBordas.XBorda[k] = -999.0F;
        CoordBordas.YBorda[k] = -999.0F;
      }
    }

  // Normalização da Matriz de Magnitudes G

  Gmax = 0.0F;
  for (i=0; i<sImg; i++)
    if (G[i] > Gmax)
      Gmax = G[i];
  for (i=0; i<sImg; i++)
    G[i] = G[i]/Gmax*255;

  // Limiarização da matriz normalizada de magnitudes dos gradientes (segundo J. F. Haddon)

  limiar = CalcLimiarHadd(pcFBordas, sqrt(2.0)*255/Gmax);
  for (i=0; i<sImg; i++)
    if (G[i] <= limiar) {
      G[i] = 0.0F;
      CoordBordas.XBorda[i] = -999.0F;
      CoordBordas.YBorda[i] = -999.0F;
    }

  // Afinamento das Bordas

  hM3 = GlobalAlloc(GHND, (DWORD)(sImg));
  imgBFinas = (LPBYTE) GlobalLock(hM3);
  AfinaBordas(imgBFinas, nImgOut, G, D, Rec);
  for (i=0; i<sImg; i++)
    if ((imgBFinas[i] > 0) && (Bin))
      imgBFinas[i] = 255;

  // Liberação de memória

  GlobalUnlock(hM1);
  GlobalUnlock(hM2);

  // Apresentação da imagem de bordas afinadas

  ApresentaImgBordas(nCol, nLin, imgBFinas, nImgIn, nImgOut, Rec, tipBorda);

  // Liberação de memória

  GlobalUnlock(hM3);

  return;
}

/*****************************************************************************************************
 DBordasSobel
 ------------
 Detecta as bordas da imagem com os operadores de Sobel.
 Parâmetros:
   ImgIn     : Imagem original
   ImgOut    : Imagem de bordas
   tipBorda  : 0)borda escuro-claro; 1)borda claro-escuro; 2)dupla borda
   Bin       : TRUE-> binariza a imagem de bordas
   pcFBordas: porcentagem tolerada de falsas bordas
   Rec       : TRUE-> a imagem ImgIn foi reconstruída
 Variáveis Globais:
   imgRec, CoordBordas
 *****************************************************************************************************/
void DBordasSobel(int nImgIn, int nImgOut, int tipBorda, BOOL Bin, double pcFBordas, BOOL Rec)
{
  int i,j,k;                              // índices
```

```
HGLOBAL hM1,hM2,hM3;                    // manipuladores de memória
int nLin, nCol;                         // número de linhas e colunas da imagem original
int sImg;                               // tamanho da imagem
double Gx, Gy;                          // gradientes da imagem segundo x e y
float *G;                               // matriz com os valores de magnitude do gradiente da imagem
float Gmax;                             // máximo valor de G
unsigned char *D;                       // direção do vetor Gradiente: 0-> 0; 1-> 45 ... 7-> 315 (graus)
int limiar;                             // limiar de separação de bordas-ruído
double ang, ang1;                       // ângulo do vetor gradiente relativamente ao eixo x
double cAng, cAng2;                     // inclinações do vetor gradiente
float coef;                             // coeficiente aplicado às coordenadas dos pixels da imagem
LPBYTE imgBFinas;                       // imagem de bordas afinadas
float (*pfGrad)(double x,double y);     // ponteiro para função que calcula gradiente

// Cálculo dos parâmetros básicos utilizados pelo algoritmo de detecção de bordas

sImg = CalcParImgBordas(nImgIn,Rec, &nCol, &nLin, &coef);

// Seleção do método de cálculo do gradiente em função do tipo de borda

switch(tipBorda) {
  case 0: pfGrad = CalcMagGrad1; break;
  case 1: pfGrad = CalcMagGrad2; break;
  case 2: pfGrad = CalcMagGrad3; break;
}

// Alocação de memória para as matrizes gradiente G e direção D

hM1 = GlobalAlloc(GHND, (DWORD)(sImg*sizeof(float)));
G = (float*) GlobalLock(hM1);
hM2 = GlobalAlloc(GHND, (DWORD)(sImg));
D = (unsigned char *) GlobalLock(hM2);

// Cálculo da matriz Gradiente G, da matriz direção D e das coordenadas das bordas

for (i=1; i<nLin-1; i++)
  for (j=1; j<nCol-1; j++) {
    Gx = imgRec[(i-1)*nCol+j+1]-imgRec[(i-1)*nCol+j-1]+
         (imgRec[i*nCol+j+1]-imgRec[i*nCol+j-1])*2+
         imgRec[(i+1)*nCol+j+1]-imgRec[(i+1)*nCol+j-1];
    Gy = imgRec[(i+1)*nCol+j-1]-imgRec[(i-1)*nCol+j-1]+
         (imgRec[(i+1)*nCol+j]-imgRec[(i-1)*nCol+j])*2+
         imgRec[(i+1)*nCol+j+1]-imgRec[(i-1)*nCol+j+1];
    cAng = fabs(Gy/Gx);
    k = i*nCol+j;
    G[k] = pfGrad(Gx, Gy);
    if (G[k] > 0) {
      ang = atan(cAng)/GRAD;
      cAng2 = tan(ang-45);
      ang1 = ang;
      if ((ang>18.43) && (ang<=45))
        ang1 = atan((7*cAng*cAng+6*cAng-1)/(-9*cAng*cAng+22*cAng-1));
      else if ((ang>45) && (ang<=71.57))
        ang1 = 45+atan((7*cAng2*cAng2+6*cAng2-1)/(-9*cAng2+22*cAng2-1));
      D[k] = CalcOriGrad(Gx, Gy, ang1);
      CoordBordas.XBorda[k] = coef*(float)j;
      CoordBordas.YBorda[k] = coef*(float)i;
    }
    else {
      CoordBordas.XBorda[k] = -999.0F;
      CoordBordas.YBorda[k] = -999.0F;
    }
  }

// Normalização da Matriz de Magnitudes G

Gmax = 0.0F;
for (i=0; i<sImg; i++)
  if (G[i] > Gmax)
    Gmax = G[i];
for (i=0; i<sImg; i++)
  G[i] = G[i]/Gmax*255;

// Limiarização da matriz normalizada de magnitudes dos gradientes (segundo J. F. Haddon)

limiar = CalcLimiarHadd(pcFBordas, sqrt(12.0)*255/Gmax);
for (i=0; i<sImg; i++)
  if (G[i] <= limiar) {
    G[i] = 0.0F;
    CoordBordas.XBorda[i] = -999.0F;
    CoordBordas.YBorda[i] = -999.0F;
  }
// Afinamento das Bordas
```

```
  hM3 = GlobalAlloc(GHND, (DWORD)(sImg));
  imgBFinas = (LPBYTE) GlobalLock(hM3);

  AfinaBordas(imgBFinas, nImgOut, G, D, Rec);
  for (i=0; i<sImg; i++)
    if ((imgBFinas[i] > 0) && (Bin))
      imgBFinas[i] = 255;
  GlobalUnlock(hM1);
  GlobalUnlock(hM2);

  // Apresenta imagem de bordas afinadas

  ApresentaImgBordas(nCol, nLin, imgBFinas, nImgIn, nImgOut, Rec, tipBorda);
  GlobalUnlock(hM3);

  return;
}

/*****************************************************************************************************
 DBordasSubPixel5x5
 ------------------
 Detecta as bordas da imagem utilizando o método proposto por Lyvers et al.
 Parâmetros:
   ImgIn     : Imagem original
   ImgOut    : Imagem de bordas
   tipBorda  : 0)borda escuro-claro; 1)borda claro-escuro; 2)dupla borda
   Bin       : TRUE-> binariza a imagem de bordas
   pcFBordas: porcentagem tolerada de falsas bordas
   Rec       : TRUE-> a imagem ImgIn foi reconstruída
 Variáveis Globais:
   imgRec, CoordBordas
 *****************************************************************************************************/
void DBordasSubPixel5x5(int nImgIn, int nImgOut, int tipBorda, BOOL Bin, double pcFBordas, BOOL Rec)
{
  int i,j,k,m,n;          // índices
  HGLOBAL hM1,hM2,hM3;    // manipuladores de memória
  int nLin, nCol;         // número de linhas e de colunas da imagem original
  int sImg;               // tamanho da imagem
  double M00;             // momento de ordem 0 da imagem em uma janela 5x5
  double M10, M01;        // momentos x e y da imagem em uma janela 5x5
  double M11;             // produto de inércia da imagem em uma janela 5x5
  double M20, M02;        // momentos de inércia da imagem em uma janela 5x5
  double Mlin10;          // momento x da imagem em uma janela 5x5 rotacionada
  double Mlin20;          // momento de inércia da imagem em uma janela 5x5 rotacionada
  double L;               // raio vetor da borda relativamente ao centro da máscara 5x5
  double Lx,Ly;           // componentes x e y do raio vetor L
  float *G;               // matriz com os valores da magnitude das bordas da imagem
  float Gmax;             // valor máximo do gradiente de variação de nível de cinza
  float grad;             // variável auxiliar
  unsigned char *D;       // direção do vetor Gradiente: 0-> 0; 1-> 45 ... 7-> 315 (graus)
  int limiar;             // limiar de separação bordas - ruído
  double ang;             // ângulo do vetor gradiente relativamente ao eixo x
  LPBYTE imgBFinas;       // pixels da imagem de bordas
  float coef;             // coeficiente aplicado às coordenadas dos pixels da imagem
                  // Para o cálculo de área da imagem
  double M00Mask[5][5] = {{ 0.0219, 0.1231, 0.1573, 0.1231, 0.0219},
                          { 0.1231, 0.1600, 0.1600, 0.1600, 0.1231},
                          { 0.1573, 0.1600, 0.1600, 0.1600, 0.1231},
                          { 0.1231, 0.1600, 0.1600, 0.1600, 0.1231},
                          { 0.0219, 0.1231, 0.1573, 0.1231, 0.0219}};
                  // Para o cálculo de produto de inércia
  double M11Mask[5][5] = {{ 0.0098, 0.0352, 0.0000,-0.0352,-0.0098},
                          { 0.0352, 0.0256, 0.0000,-0.0256,-0.0352},
                          { 0.0000, 0.0000, 0.0000, 0.0000, 0.0000},
                          {-0.0352,-0.0256, 0.0000, 0.0256, 0.0352},
                          {-0.0098,-0.0352, 0.0000, 0.0352, 0.0098}};
                  // Para o cálculo de momento Mx
  double M10Mask[5][5] = {{-0.0147,-0.0469, 0.0000, 0.0469, 0.0147},
                          {-0.0933,-0.0640, 0.0000, 0.0640, 0.0933},
                          {-0.1253,-0.0640, 0.0000, 0.0640, 0.1253},
                          {-0.0933,-0.0640, 0.0000, 0.0640, 0.0933},
                          {-0.0147,-0.0469, 0.0000, 0.0469, 0.0147}};
                  // Para o cálculo de momento My
  double M01Mask[5][5] = {{-0.0147,-0.0933,-0.1253,-0.0933,-0.0147},
                          {-0.0469,-0.0640,-0.0640,-0.0640,-0.0469},
                          { 0.0000, 0.0000, 0.0000, 0.0000, 0.0000},
                          { 0.0469, 0.0640, 0.0640, 0.0640, 0.0469},
                          { 0.0147, 0.0933, 0.1253, 0.0933, 0.0147}};
                  // Para o cálculo de momento de inércia x
  double M20Mask[5][5] = {{ 0.0099, 0.0194, 0.0021, 0.0194, 0.0099},
                          { 0.0719, 0.0277, 0.0021, 0.0277, 0.0719},
                          { 0.1019, 0.0277, 0.0021, 0.0277, 0.1019},
                          { 0.0719, 0.0277, 0.0021, 0.0277, 0.0719},
                          { 0.0099, 0.0194, 0.0021, 0.0194, 0.0099}};
```

```
                // Para o cálculo de momento de inércia y
double M02Mask[5][5] = {{ 0.0099, 0.0719, 0.1019, 0.0719, 0.0099},
                        { 0.0194, 0.0277, 0.0277, 0.0277, 0.0194},
                        { 0.0021, 0.0021, 0.0021, 0.0021, 0.0021},
                        { 0.0194, 0.0277, 0.0277, 0.0277, 0.0194},
                        { 0.0099, 0.0719, 0.1019, 0.0719, 0.0099}};
                // ponteiro para função que calcula gradiente
float (*pfGrad)(double x,double y);

// Cálculo dos parâmetros básicos utilizados pelo algoritmo de detecção de bordas

sImg = CalcParImgBordas(nImgIn, Rec, &nCol, &nLin, &coef);

// Seleção do método de cálculo do gradiente em função do tipo de borda

switch(tipBorda) {
  case 0: pfGrad = CalcMagGrad1; break;
  case 1: pfGrad = CalcMagGrad2; break;
  case 2: pfGrad = CalcMagGrad3; break;
}

// Alocação de memória para as matrizes gradiente G e direção D

hM1 = GlobalAlloc(GHND, (DWORD)(sImg*sizeof(float)));
G = (float*) GlobalLock(hM1);
hM2 = GlobalAlloc(GHND, (DWORD)(sImg));
D = (unsigned char *) GlobalLock(hM2);

// Cálculo da matriz Gradiente G, da matriz direção D e das coordenadas das bordas

for (i=0; i<nLin-2; i++)
  for (j=2; j<nCol; j++) {
    M00 = 0.0;
    M10 = 0.0;
    M01 = 0.0;
    M11 = 0.0;
    M20 = 0.0;
    M02 = 0.0;
    for (m=-2; m<=2; m++)
      for (n=-2; n<=2; n++) {
        k = (i+m)*nCol+(j+n);
        M00+= imgRec[k]*M00Mask[m+2][n+2];
        M10+= imgRec[k]*M10Mask[m+2][n+2];
        M01+= imgRec[k]*M01Mask[m+2][n+2];
        M11+= imgRec[k]*M11Mask[m+2][n+2];
        M20+= imgRec[k]*M20Mask[m+2][n+2];
        M02+= imgRec[k]*M02Mask[m+2][n+2];
      }

    // Determinação da magnitude e orientação do gradiente

    k = i*nCol+j;
    Mlin10 = sqrt(M10*M10+M01*M01);
    Mlin20 = (M10*M10*M20+2.0*M01*M10*M11+M01*M01*M02)/(Mlin10*Mlin10);
    L = (4.0*Mlin20-M00)/(3.0*Mlin10);
    Lx = L*M10/Mlin10;
    Ly = L*M01/Mlin10;
    if ((fabs(Lx)<=0.5) && (fabs(Ly)<=0.5)) { // a borda se localiza no interior do pixel central
      //G[k] = (float)(1.5*pfGrad(M10,M01)/sqrt(pow(1.0-L*L,3)));
      grad = (float)(1.5*Mlin10/sqrt(pow(1.0-L*L,3)));
      switch (tipBorda) {
        case 0:           // Borda escuro-clara
          if (fabs(M10) > fabs(M01))
            { G[k] = (M10>0) ? grad : 0.0F; }
          else
            { G[k] = (M01>0) ? grad : 0.0F; }
          break;
        case 1:           // Borda claro-escura
          if (fabs(M10) > fabs(M01))
            { G[k] = (M10<0) ? grad : 0.0F; }
          else
            { G[k] = (M01<0) ? grad : 0.0F; }
          break;
        case 2:           // Dupla borda
          G[k] = grad;
          break;
      }
      if (G[k]!=0) {
        ang = fabs(atan(M01/M10))/GRAD;
        D[k] = CalcOriGrad(M10, M01, ang);
        CoordBordas.XBorda[k] = (float)(j+Lx);
        CoordBordas.YBorda[k] = (float)(i+Ly);
      }
      else {
```

```
          CoordBordas.XBorda[k] = -999.0F;
          CoordBordas.YBorda[k] = -999.0F;
        }
      }
      else
        G[k] = 0.0F;
    }

  // Normalização da Matriz de Magnitudes G

  Gmax = 0.0F;
  for (i=0; i<sImg; i++)
    if (G[i] > Gmax)
      Gmax = G[i];
  for (i=0; i<sImg; i++)
    G[i] = G[i]/Gmax*255;

  // Limiarização da matriz normalizada de magnitudes dos gradientes (segundo J. F. Haddon)

  limiar = CalcLimiarHadd(pcFBordas, sqrt(0.1005)*255/Gmax);
  for (i=0; i<sImg; i++)
    if (G[i] <= limiar) {
      G[i] = 0.0F;
      CoordBordas.XBorda[i] = -999.0F;
      CoordBordas.YBorda[i] = -999.0F;
    }

  // Afinamento das Bordas

  hM3 = GlobalAlloc(GHND, (DWORD)(sImg));
  imgBFinas = (LPBYTE) GlobalLock(hM3);
  AfinaBordas(imgBFinas, nImgOut, G, D, Rec);
  for (i=0; i<sImg; i++)
    if ((imgBFinas[i] > 0) && (Bin))
      imgBFinas[i] = 255;
  GlobalUnlock(hM1);
  GlobalUnlock(hM2);

  // Apresenta imagem de bordas afinadas

  ApresentaImgBordas(nCol, nLin, imgBFinas, nImgIn, nImgOut, Rec, tipBorda);
  GlobalUnlock(hM3);

  return;
}

/*****************************************************************************************************
 DBordasSubPixel3x3
 ------------------
 Detecta as bordas da imagem utilizando o método proposto por Jensen et Anastassiou.
 Parâmetros:
   ImgIn    : Imagem original
   ImgOut   : Imagem de bordas
   tipBorda : 0)borda escuro-claro; 1)borda claro-escuro; 2)dupla borda
   Bin      : TRUE-> binariza a imagem de bordas
   pcFBordas: porcentagem tolerada de falsas bordas
   Rec      : TRUE-> a imagem ImgIn foi reconstruída
 Variáveis Globais:
   imgRec, CoordBordas
 *****************************************************************************************************/
void DBordasSubPixel3x3(int nImgIn, int nImgOut, int tipBorda, BOOL Bin, double pcFBordas, BOOL Rec)
{
  int i,j,k,m,n;          // índices
  HGLOBAL hM1,hM2,hM3;    // manipuladores de memória
  int nLin, nCol;         // número de linhas e colunas da imagem original
  int sImg;               // tamanho da imagem
  double Lambda1;         // convolução da imagem com a máscara M1
  double Lambda2;         // convolução da imagem com a máscara M2
  double Lambda3;         // convolução da imagem com a máscara M3
  double Lambda4;         // convolução da imagem com a máscara M4
  double Rho;             // distância do centro do pixel à borda
  double dx,dy;           // projeções x,y, da distância Rho
  double sigma;           // semi-arco capaz da borda relativamente a um círculo de raio R=(1+sqrt(2))/2
  float *G;               // matriz com os valores de magnitude do gradiente da imagem
  float Gmax;             // máximo valor de G
  float grad;             // variável auxiliar
  unsigned char *D;       // direção do vetor Gradiente: 0-> 0; 1-> 45 ... 7-> 315 (graus)
  int limiar;             // limiar de separação de bordas-ruído
  double R;               // raio da região onde se localiza a borda
  double c,s;             // cosseno e seno de um dado ângulo
  double ang;             // ângulo do vetor gradiente relativamente ao eixo x
  float coef;             // coeficiente aplicado às coordenadas dos pixels da imagem
  LPBYTE imgBFinas;       // pixels da imagem de bordas
  float (*pfGrad)(double x,double y); // ponteiro para função que calcula gradiente
```

```
                        // máscaras
double M1[3][3] = {{-0.31333, 0       , 0.31333},
                   {-0.44311, 0       , 0.44311},
                   {-0.31333, 0       , 0.31333}};
double M2[3][3] = {{-0.31333,-0.44311,-0.31333},
                   { 0       , 0       , 0       },
                   { 0.31333, 0.44311, 0.31333}};
double M3[3][3] = {{ 0       ,-0.44311, 0       },
                   { 0.44311, 0       , 0.44311},
                   { 0       ,-0.44311, 0       }};
double M4[3][3] = {{-0.44311, 0       , 0.44311},
                   { 0       , 0       , 0       },
                   { 0.44311, 0       ,-0.44311}};

// Cálculo dos parâmetros básicos utilizados pelo algoritmo de detecção de bordas

sImg = CalcParImgBordas(nImgIn, Rec, &nCol, &nLin, &coef);

// Seleção do método de cálculo do gradiente em função do tipo de borda

switch(tipBorda) {
  case 0: pfGrad = CalcMagGrad1; break;
  case 1: pfGrad = CalcMagGrad2; break;
  case 2: pfGrad = CalcMagGrad3; break;
}

// Alocação de memória para as matrizes gradiente G e direção D

hM1 = GlobalAlloc(GHND, (DWORD)(sImg*sizeof(float)));
G = (float*) GlobalLock(hM1);
hM2 = GlobalAlloc(GHND, (DWORD)(sImg));
D = (unsigned char *) GlobalLock(hM2);

// Cálculo da matriz Gradiente G, da matriz direção D e das coordenadas das bordas

R = (1.0+sqrt(2.0))*0.5;
for (i=0; i<nLin-1; i++)
  for (j=1; j<nCol; j++) {
    Lambda1 = 0.0;
    Lambda2 = 0.0;
    Lambda3 = 0.0;
    Lambda4 = 0.0;
    for (m=-1; m<=1; m++)
      for (n=-1; n<=1; n++) {
        k = (i+m)*nCol+(j+n);
        Lambda1+= imgRec[k]*M1[m+1][n+1];
        Lambda2+= imgRec[k]*M2[m+1][n+1];
        Lambda3+= imgRec[k]*M3[m+1][n+1];
        Lambda4+= imgRec[k]*M4[m+1][n+1];
      }
    k = i*nCol+j;
    // Determina a orientação da borda relativamente aos eixos X-Y da imagem
    ang = atan2(Lambda2,Lambda1);
    c = cos(ang);
    s = sin(ang);
    // Calcula a distância Rho (e suas projeções) entre a borda e o pixel-referência
    Rho = R*(Lambda3*cos(2*ang)+Lambda4*sin(2*ang))/(Lambda1*c+Lambda2*s);
    Rho = fabs(Rho);
    dx  = Rho*c;
    dy  = Rho*s;
    // Se a borda se localiza no interior do pixel-referência calcula seu gradiente e posição
    if ((fabs(dx)<=0.5) && (fabs(dy)<=0.5)) {
      // Determina o ângulo sigma: semi-arco-capaz que contém a borda distante Rho do pixel-referência
      sigma = acos(fabs(Rho)/R);
      // Determina a magnitude do gradiente
      grad = (float)(sqrt(PI)*fabs((Lambda1*c+Lambda2*s)/(2.0*sin(sigma))));
      // Determina a localização da borda
      switch (tipBorda) {
        case 0:           // Borda escuro-clara
          if (fabs(Lambda1) > fabs(Lambda2))
            { G[k] = (Lambda1>0) ? grad : 0.0F; }
          else
            { G[k] = (Lambda2>0) ? grad : 0.0F; }
          break;
        case 1:           // Borda claro-escura
          if (fabs(Lambda1) > fabs(Lambda2))
            { G[k] = (Lambda1<0) ? grad : 0.0F; }
          else
            { G[k] = (Lambda2<0) ? grad : 0.0F; }
          break;
        case 2:           // Dupla borda
          G[k] = grad;
          break;
      }
```

```
        // Determina a localização da borda
        if (G[k]!=0) {
          CoordBordas.XBorda[k] = (float)(j+dx);
          CoordBordas.YBorda[k] = (float)(i+dy);
          ang = atan(fabs(Lambda2/Lambda1))/GRAD;
          D[k] = CalcOriGrad(Lambda1, Lambda2, ang);
        }
        else {
          CoordBordas.XBorda[k] = -999.0F;
          CoordBordas.YBorda[k] = -999.0F;
        }
      }
    }

  // Normalização da Matriz de Magnitudes G

  Gmax = 0.0F;
  for (i=0; i<sImg; i++)
    if (G[i] > Gmax)
      Gmax = G[i];
  for (i=0; i<sImg; i++)
    G[i] = G[i]/Gmax*255;

  // Limiarização da matriz normalizada de magnitudes dos gradientes (segundo J. F. Haddon)

  limiar = CalcLimiarHadd(pcFBordas, sqrt(0.7854)*255/Gmax);
  for (i=0; i<sImg; i++)
    if (G[i] <= limiar) {
      G[i] = 0.0F;
      CoordBordas.XBorda[i] = -999.0F;
      CoordBordas.YBorda[i] = -999.0F;
    }

  // Afinamento das Bordas

  hM3 = GlobalAlloc(GHND, (DWORD)(sImg));
  imgBFinas = (LPBYTE) GlobalLock(hM3);
  AfinaBordas(imgBFinas, nImgOut, G, D, Rec);
  for (i=0; i<sImg; i++)
    if ((imgBFinas[i] > 0) && (Bin))
      imgBFinas[i] = 255;
  GlobalUnlock(hM1);
  GlobalUnlock(hM2);

  // Apresenta imagem de bordas afinadas

  ApresentaImgBordas(nCol, nLin, imgBFinas, nImgIn, nImgOut, Rec, tipBorda);
  GlobalUnlock(hM3);

  return;
}

/*****************************************************************************************************
 CalcParImgBordas
 ----------------
 Determina parâmetros básicos usados na geração da imagem de bordas.
 Parâmetros:
   nImgIn: número da imagem original
   Rec   : TRUE-> imagem reconstruída
   nCol  : número de colunas da imagem de bordas
   nLin  : número de linhas da imagem de bordas
   coef  : 1-> imagem normal; 0.5-> imagem reconstruída
 *****************************************************************************************************/
int CalcParImgBordas(int nImgIn, BOOL Rec, int *nCol, int *nLin, float *coef)
{
  if (Rec) {
    *nCol = 2*DimBitmap[nImgIn].cx;
    *nLin = 2*DimBitmap[nImgIn].cy;
    *coef = 0.5F;
  }
  else {
    *nCol = DimBitmap[nImgIn].cx;
    *nLin = DimBitmap[nImgIn].cy;
    *coef = 1.0F;
  }

  return (*nCol*(*nLin));
}

/*****************************************************************************************************
 CalcMagGrad1
 ------------
 Calcula a magnitude do gradiente para o caso de bordas escuro-claras.
 Parâmetros:
```

```
  Gx,Gy: componentes x e y do gradiente da imagem
 Retorna:
  magnitude do gradiente
 ********************************************************************************************************************/
float CalcMagGrad1(double Gx, double Gy)
{
  double G;
  if (fabs(Gx) > fabs(Gy)) {
    G = (Gx>0) ? sqrt(Gx*Gx+Gy*Gy) : 0; }
  else {
    G = (Gy>0) ? sqrt(Gx*Gx+Gy*Gy) : 0; }
  return (float)G;
}

/********************************************************************************************************************
 CalcMagGrad2
 ------------
 Calcula a magnitude do gradiente para o caso de bordas claro-escuras.
 Parâmetros:
    Gx,Gy: componentes x e y do gradiente da imagem
 Retorna:
      magnitude do gradiente
 ********************************************************************************************************************/
float CalcMagGrad2(double Gx, double Gy)
{
  double G;
  if (fabs(Gx) > fabs(Gy)) {
    G = (Gx<0) ? sqrt(Gx*Gx+Gy*Gy) : 0; }
  else {
    G = (Gy<0) ? sqrt(Gx*Gx+Gy*Gy) : 0; }
  return (float)G;
}

/********************************************************************************************************************
 CalcMagGrad1
 ------------
 Calcula a magnitude do gradiente para o caso de duplas bordas.
 Parâmetros:
   Gx,Gy: componentes x e y do gradiente da imagem
 Retorna:
   magnitude do gradiente
 ********************************************************************************************************************/
float CalcMagGrad3(double Gx, double Gy)
{
  return ((float)sqrt(Gx*Gx+Gy*Gy));
}

/********************************************************************************************************************
 CalcOriGrad
 -----------
 Classifica o vetor gradiente em 8 categorias segundo orientações de 0 a 360 graus, ou seja: 0, 45, 90,
 135, 180, 225, 270, 315 graus.
 Parâmetros:
   Gx, Gy:  componentes x, y do gradiente da imagem
   ang    :  ângulo formado pelo gradiente com o eixo x
 Retorna:
   código do vetor -> 0 a 7
 ********************************************************************************************************************/
unsigned char CalcOriGrad(double Gx, double Gy, double ang)
{
  unsigned char D; // códigos (0-7) de ângulos espaçados de 45 graus

  if (ang<22.5)
    {D = (Gx>0) ? 0 : 4;}   // vetor gradiente a 0 e 180 graus
  else if (ang>77.5)
    {D = (Gy>0) ? 6 : 2;}   // vetor gradiente a 270 e 90 graus
  else {
    if (Gx>0)
      {D = (Gy<0) ? 1 : 7;} // vetor gradiente a 315 e 45 graus
    else
      {D = (Gy<0) ? 3 : 5;} // vetor gradiente a 135 e 225 graus
  }

  return D;
}

/********************************************************************************************************************
 CalcLimiarHadd
 --------------
 Calcula o valor ótimo de limiar para uma imagem de bordas utilizando a formulação proposta por J.Haddon.
 Parâmetros:
   pcFBordas: % aceitável de bordas falsas
   fator    : contribuição dos pesos do detector de bordas
 Retorna:
```

```
   limiar: valor do limiar
 ***************************************************************************************************************/
int CalcLimiarHadd(double pcFBordas, double fator)
{
  int limiar;   // valor ótimo do limiar

  limiar = (int)(sqrt(-2*log(pcFBordas/2))*SigmaN*fator);

  // Obs: SigmaN é o desvio-padrão do ruído Gaussiano da imagem

  return limiar;
}

/***************************************************************************************************************
 AfinaBordas
 -----------
 Executa o afinamento das bordas, segundo o método proposto por Nevatia e Babu.
 Parâmetros:
   imgBordas: pixels da imagem de bordas afinadas
   nImgOri  : número da imagem a ser apresentada
   G        : magnitudes das bordas
   D        : orientações das bordas
   Dup      : TRUE-> a imagem é de dupla resolução
 ***************************************************************************************************************/
void AfinaBordas(LPBYTE imgBordas, int nImg, float *G, unsigned char *D, BOOL Dup)
{
  int nLin, nCol;   // número de linhas e de colunas da imagem
  int m[4], n[4];   // vetores de orientação
  int i,j,k,p;      // índices
  int f0,f1,f2;     // no. de pixels da imagem anteriores às linhas i, i-1 e i+1

  // Recuperação das informações básicas da imagem original

  if (!Dup) {
    nLin = DimBitmap[nImg].cy;
    nCol = DimBitmap[nImg].cx;
  }
  else { // imagem reconstruída, com dupla resolução
    nLin = 2*DimBitmap[nImg].cy;
    nCol = 2*DimBitmap[nImg].cx;
  }

  // Algoritmo de afinamento aplicado à matriz de gradientes G

  for (i=1; i<nLin-1; i++)
    for (j=1; j<nCol-1; j++) {
      k = i*nCol+j;
      if (G[k] > 0) {
        f0   = i*nCol;
        f1   = f0-nCol;
        f2   = f0+nCol;
        m[0] = f0+j+1;
        m[1] = f1+j+1;
        m[2] = f1+j;
        m[3] = f1+j-1;
        n[0] = f0+j-1;
        n[1] = f2+j-1;
        n[2] = f2+j;
        n[3] = f2+j+1;
        p = D[k]%4;
        if ((G[k] >= G[m[p]]) && (G[k] >= G[n[p]]))
          imgBordas[k] = (BYTE)G[k];
        else {
          CoordBordas.XBorda[k] = -999.0F;
          CoordBordas.YBorda[k] = -999.0F;
        }
      }
    }

  return;
}

/***************************************************************************************************************
 ApresentaImgBordas
 ------------------
 Apresenta a imagem de bordas. Caso esta seja de dupla resolução, são apresentadas 4 sub-imagens, uma para
 cada um dos quadrantes.
 Parâmetros:
   nCol2    : número de colunas da imagem de bordas
   nLin2    : número de linhas da imagem de bordas
   imgBordas: imagem de bordas
   nImgIn   : número da imagem original
   nImgOut  : número da imagem apresentada
   Rec      : TRUE-> se a imagem de bordas é de dupla resolução
```

```
   tB       : 0)borda escuro-claro; 1)borda claro-escuro; 2)dupla borda
 ***************************************************************************************************************/
void ApresentaImgBordas(int nCol2, int nLin2, LPBYTE imgBordas, int nImgIn, int nImgOut, BOOL Rec, int tB)
{
  int i,j,a,b;       // índices
  int nCol, nLin;    // número de colunas e de linhas da imagem de baixa resolução
  int sImg;          // tamanho da imagem de baixa resolução
  HGLOBAL hM;        // manipulador de memória
  LPBYTE imgA;       // imagem apresentada

  // Alocação de memória

  hM = GlobalAlloc(GHND, (DWORD)(nCol2*nLin2));
  imgA = (LPBYTE) GlobalLock(hM);

  // Definição do título

  if (Rec) { // Apresentação de imagem com dupla resolução

    nCol = nCol2/2;
    nLin = nLin2/2;
    sImg = nCol*nLin;

    // Apresenta a imagem de bordas em alta resolução, através de 4 sub-imagens

    for (a=0; a<=nLin; a+=nLin)
      for (b=0; b<=nCol; b+=nCol) {
        for (i=0; i<nLin; i++)
          for (j=0; j<nCol; j++) {
            PixImg[nImgOut][i*nCol+j] = imgBordas[(i+a)*nCol*2+j+b];
            imgA[i*nCol+j] = ConvIndPal[PixImg[nImgOut][i*nCol+j]];
          }
        SetBitmapBits(hbm[nImgOut], (DWORD)sImg, (LPSTR)imgA);
        sprintf(NomeImg[nImgOut], "Imagem de bordas: referência (linha %d, coluna %d)", a, b);
        MostraImagem(nImgOut);
        Sleep(2000);
      }

    // Apresenta a imagem de bordas em baixa resolução

    for (i=0; i<sImg; i++)
      PixImg[nImgOut][i] = 0;
    for (i=0; i<2*nLin; i++)
      for (j=0; j<2*nCol; j++)
        if (imgBordas[i*2*nCol+j]==255)
          PixImg[nImgOut][(i/2)*nCol+j/2] = 255;
    switch (tB) {
      case 0: sprintf(NomeImg[nImgOut],"Bordas Escuro->Claras"); break;
      case 1: sprintf(NomeImg[nImgOut],"Bordas Claro->Escuras"); break;
      case 2: sprintf(NomeImg[nImgOut],"Bordas Duplas"); break;
    }
    SetBitmapBits(hbm[nImgOut], (DWORD)sImg, (LPSTR)PixImg[nImgOut]);
    MostraImagem(nImgOut);
  }

  else { // Apresentação de imagem com baixa resolução

    sImg = nCol2*nLin2;
    for (i=0; i<sImg; i++) {
      PixImg[nImgOut][i] = imgBordas[i];
      imgA[i] = ConvIndPal[imgBordas[i]];
    }
    switch (tB) {
      case 0: sprintf(NomeImg[nImgOut],"Bordas(escuro-claras) da Imagem %d", nImgIn); break;
      case 1: sprintf(NomeImg[nImgOut],"Bordas(claro-escuras) da Imagem %d", nImgIn); break;
      case 2: sprintf(NomeImg[nImgOut],"Bordas(duplas) da Imagem %d", nImgIn); break;
    }
    SetBitmapBits(hbm[nImgOut], (DWORD)sImg, (LPSTR)imgA);
    MostraImagem(nImgOut);
  }

  // Liberação de memória

  GlobalUnlock(hM);

  return;
}

/***************************************************************************************************************
 MedOrdpRot
 ----------
 Gera máscaras rotuladas para medição em escala ordinal.
 Parâmetros:
   nImg1: número da imagem binária original
```

```
   nImg2: número da imagem da máscara rotulada
   nImg3: número da imagem da máscara binária
 ***************************************************************************************************************/
int MedOrdpRot(int nImg1, int nImg2, int nImg3)
{
  int sImg;          // tamanho da imagem
  int n;             // tamanho máximo do elemento estruturante
  int np;            // tamanho real do elemento estruturante
  ponto *ElemEst;    // elemento estruturante
  HGLOBAL hM;        // manipulador de memória
  int nObj;          // número de objetos rotulados

  #define raio 1

  sImg = DimBitmap[nImg1].cx*DimBitmap[nImg1].cy;
  n = (int)((float)((raio+1)*(raio+1)*PI));
  hM = GlobalAlloc(GHND, (DWORD)(n*sizeof(ponto)));
  ElemEst = (ponto *) GlobalLock(hM);
  np = CriaElemEstCirc(raio, ElemEst);
  ApagaImagem(nImg2);

  // Construção de máscara com objetos rotulados

  Dilatacao(nImg1, nImg3, np, ElemEst);
  nObj = Rotulacao(nImg3, nImg2, 0.3);

  #undef raio

  // Liberação de memória

  GlobalUnlock(hM);

  return nObj;
}

/***************************************************************************************************************
 TransHough
 ----------
 Gera a transformada de Hough da imagem e, em seguida, busca máximos locais no espaço de Hough para
 caracterizar as retas da imagem.
 Parâmetros:
   nImgIn   : número da imagem original
   nImgOut  : número da imagem de retas
   Dup      : TRUE-> imagem em dupla resolução
   imgBin   : TRUE-> imagem binária; FALSE-> imagem em níveis de cinza
   aut      : TRUE-> procedimento automático
   ThetaMin: ângulo mínimo (graus) do espaço de Hough
   ThetaMax: ângulo máximo (graus) do espaço de Hough
   dTheta   : quantização angular do espaço de Hough
   L        : comprimento mínimo das retas
   b        : espalhamento dos pontos em torno da reta central
   RetPar   : TRUE-> retas paralelas; FALSE->retas quaisquer
   dPic     : distância mínima entre picos no espaço de Hough
   nfxPic   : TRUE-> localiza número fixo de picos no espaço de Hough
   nPic     : número de picos no espaço de Hough
   Fcorte   : limiar no espaço de Hough
 ***************************************************************************************************************/
void TransHough(int nImgIn,int nImgOut,BOOL Dup,BOOL imgBin,BOOL aut,double ThetaMin,double ThetaMax,
                double dTheta,double L,double b,BOOL RetPar,int dPic,BOOL nfxPic,int nPic,double Fcorte)
{
  int i,j;                    // índices
  HGLOBAL hM1, hM2;           // manipuladores de memória
  double cprop;               // constante para representação do espaço de Hough
  int sImg;                   // tamanho da imagem em bytes
  double *H;                  // matriz de Hough
  LPBYTE imgA;                // imagem apresentada
  int nTheta;                 // número de linhas (valores Theta) da matriz de Hough
  int nRho;                   // número de colunas (valores Rho) da matriz de Hough
  double RhoMin, RhoMax;      // valores mínimo e máximo da coordenada Rho, no espaço de Hough
  double dRho;                // quantização Rho no espaço de Hough
  float X[N_PixelsX*8];       // abcissas dos pixels em torno de cada reta
  float Y[N_PixelsY*8];       // ordenadas dos pixels em torno de cada reta
  double ang;                 // orientação de uma dada reta
  int npt;                    // no. de pontos de uma dada nuvem de pixels
  double d;                   // distância de um pixel à origem
  double sigma;               // variância da distribuição de pixels em torno de uma dada reta

  // Cálculo dos limites [RhoMin,RhoMax] e da quantização dRho do espaço de Hough

  CalcParamTHough(nImgIn, ThetaMin, ThetaMax, dTheta, L, b, &nTheta, &nRho, &RhoMin, &RhoMax, &dRho);

  // Criação e acumulação da matriz de Hough

  hM1 = GlobalAlloc(GHND, (DWORD)(nTheta*nRho*sizeof(double)));
```

```
  H = (double*) GlobalLock(hM1);
  hM2 = GlobalAlloc(GHND, (DWORD)(nTheta*nRho*sizeof(double)));
  imgA = (LPBYTE) GlobalLock(hM2);
  AcumulaHough(nImgIn, imgBin, H, nTheta, ThetaMin, dTheta, nRho, RhoMin, RhoMax);

  // Criação e apresentação da imagem do espaço de Hough

  hbm[nImgOut] = CreateBitmap(nRho, nTheta, 1, 8,(VOID *)NULL);
  DimBitmap[nImgOut].cx = nRho;
  DimBitmap[nImgOut].cy = nTheta;
  sprintf(NomeImg[nImgOut],"Img%d: (Espaço de Hough)", nImgOut);
  IndImgSel = nImgOut;
  cprop = 255.0/MaxVetor(nTheta*nRho, H);
  for (i=0; i<nTheta*nRho; i++) {
    PixImg[IndImgSel][i] = (BYTE)(H[i]*cprop);
    imgA[i] = ConvIndPal[(BYTE)(H[i]*cprop)];
  }
  SetBitmapBits(hbm[nImgOut], (DWORD)(nTheta*nRho), (LPSTR)imgA);
  MostraImagem(nImgOut);

  // Encerramento opcional do processo de detecção de retas

  if (!aut)
    if (MessageBox(NULL, "Continua?", "Detecção de Retas", MB_YESNO)==IDNO) {
      GlobalUnlock(hM1);
      return;
    }

  // Determinação do número de retas e das equações das retas em coordenadas polares

  sImg = DimBitmap[nImgIn].cx*DimBitmap[nImgIn].cy;
  ApagaImagem(nImgOut);
  if (RetPar)
    AchaRetasParalelas(nImgOut,H,nTheta,ThetaMin,dTheta,nRho,RhoMin,dRho,dPic,nfxPic,nPic,Fcorte);
  else
    AchaRetas(nImgOut,H,nTheta,ThetaMin,dTheta,nRho,RhoMin,dRho,dPic,nfxPic,nPic,Fcorte);

  // Estimação do desvio-padrão da média da distribuição de pontos em torno das retas localizadas

  if (imgBin)
    for (i=0; i<FeixeRetas[nImgOut].nRetasImg; i++) {
      npt = SelecionaPontos(nImgIn,Dup,1.5,FeixeRetas[nImgOut].Raio[i],FeixeRetas[nImgOut].Angulo[i],X,Y);
      sigma = 0;
      ang = FeixeRetas[nImgOut].Angulo[i]*GRAD;
      for (j=0; j<npt; j++) {
        d = fabs(X[j]*cos(ang)+Y[j]*sin(ang))-fabs(FeixeRetas[nImgOut].Raio[i]);
        sigma+= d*d;
      }
      sigma = sigma/(double)(npt-1);
      if (npt==0) MessageBox(NULL, "Número nulo de pontos", NULL, MB_OK);
      FeixeRetas[nImgOut].Sigma[i] = sqrt(sigma/(double)npt);
      FeixeRetas[nImgOut].nPts [i] = npt;
    }
  else
    for (i=0; i<FeixeRetas[nImgOut].nRetasImg; i++)
      FeixeRetas[nImgOut].Sigma[i] = 1.0;

  // Apresentação da imagem de retas

  sprintf(NomeImg[nImgOut], "No. de Retas Localizadas via T-H: %d", FeixeRetas[nImgOut].nRetasImg);
  ImagemdeRetas(nImgIn, nImgOut, FeixeRetas[nImgOut].nRetasImg, FeixeRetas[nImgOut].Angulo, FeixeRetas[nImgOut
].Raio);

  // Liberação de Memória

  GlobalUnlock(hM1);
  GlobalUnlock(hM2);

  return;
}

/*****************************************************************************************************
 CalcParamTHough
 ---------------
 Calcula os limites da coordenada radial (RhoMin, RhoMax) e o incremento radial (dRho) em função dos
 limites da coordenada angular (ThetaMin, ThetaMax) e do correspondente incremento (dTheta). A formulação
 utilizada é a proposta por Niblack et Petkovic(1990).
 Parâmetros:
   nImgIn  : número da imagem
   ThetaMin: valor mínimo da coordenada angular
   ThetaMax: valor máximo da coordenada angular
   dTheta  : incremento da coordenada angular
   L       : comprimento da menor reta contribuinte do espaço de Hough
   b       : espalhamento das retas no espaço de Hough
```

```
   nTheta   : dimensão da matriz de acumulação na direção angular
   nRho     : dimensão da matriz de acumulação na direção radial
   RhoMin   : valor mínimo da coordenada radial
   RhoMax   : valor máximo da coordenada radial
   dRho     : incremento da coordenada radial
 ********************************************************************************************************/
void CalcParamTHough(int nImgIn, double ThetaMin, double ThetaMax, double dTheta, double L, double b,
                     int *nTheta, int *nRho, double *RhoMin, double *RhoMax, double *dRho)
{
  int nCol, nLin;    // número de colunas e de linhas da imagem nImgIn
  double Theta;      // coordenada angular do espaço de Hough
  double dAng;       // incremento da coordenada angular
  double r1,r2;      // variáveis auxiliares pará determinação de RhoMin e RhoMax
  double dr1,dr2;    // incrementos da coordenada radial

  #define sRho 6     // Espalhamento do pico de H ao longo das linhas Theta = const

  // Determinação do intervalo de variação [RhoMin,RhoMax] da coordenada radial Rho

  nCol = DimBitmap[nImgIn].cx;
  nLin = DimBitmap[nImgIn].cy;
  dAng = ThetaMax-ThetaMin;
  *RhoMin = 1000.0;
  *RhoMax =-1000.0;
  Theta   = ThetaMin;
  while (Theta<=ThetaMax) {
    if (Theta>=0.0) {
      r1 = 0.0;
      r2 = (double)nLin*sin(Theta*GRAD)+(double)nCol*cos(Theta*GRAD);
    }
    else {
      r1 = (double)nLin*sin(Theta*GRAD);
      r2 = (double)nCol*cos(Theta*GRAD);
    }
    if (r1<*RhoMin) *RhoMin = r1;
    if (r2>*RhoMax) *RhoMax = r2;
    Theta+= dAng;
  }

  // Determinação do incremento dRho da coordenada radial Rho

  dr1 = (L*sin(GRAD*dTheta*0.5)+2.0*b*cos(GRAD*dTheta*0.5))/(sRho-2.0);
  dr2 = L*sin(GRAD*dTheta);
  *dRho = __min(dr1,dr2);

  // Determinação das dimensões da matriz de acumulação

  *nTheta = (int)ceil((ThetaMax-ThetaMin)/dTheta+1);
  *nRho   = (int)((*RhoMax-*RhoMin)/(*dRho))+1;

  #undef sRho

  return;
}

/********************************************************************************************************
 AcumulaHough
 ------------
 Gera a matriz de Hough através de um processo de acumulação específico para cada tipo de imagem. No caso
 de imagens binárias, cada pixel "branco" contribuinte de uma posição do espaço de Hough implica no
 acréscimo de um valor unitário, distribuido entre as duas células vizinhas mais próximas da matriz. Para
 imagens em níveis de cinza, cada pixel contribuinte de uma dada posição do espaço de Hough acrescenta o
 seu nível de cinza à célula mais próxima. Nesse último caso, os valores de cada célula da matriz são
 normalizados mediante a divisão pelo correspondente número de pixels contribuintes.
 Parâmetros:
   nImgIn   : número da imagem original
   imgBin   : TRUE-> imagem binária FALSE-> imagem em tons de cinza
   H        : matriz de Hough
   nTheta   : dimensão de H ao longo da direção da coordenada angular
   ThetaMin: valor mínimo da coordenada angular
   dTheta   : incremento angular
   nRho     : dimensão de H ao longo da direção da coordenada radial
   RhoMin   : valor mínimo da coordenada radial
   RhoMax   : valor máximo da coordenada radial
 ********************************************************************************************************/
void AcumulaHough(int nImgIn, BOOL imgBin, double *H, int nTheta, double ThetaMin, double dTheta, int nRho,
                  double RhoMin, double RhoMax)
{
  int i,j,k,l,m;          // índices
  int nLin, nCol;         // número de linhas e de colunas da imagem nImgIn
  HGLOBAL hM;             // manipulador de memória
  double Theta;           // coordenada angular do espaço de Hough
  double Rho;             // coordenada radial do espaço de Hough
  int sImg;               // tamanho da imagem do espaço de Hough
```

```
  int *Cont;            // vetor para contagem de pixels contribuintes de uma célula da matriz H
  double cprop;         // constante de proporcionalidade
  int maxCont;          // valor máximo do vetor Cont;

  #define w 2

  nCol = DimBitmap[nImgIn].cx;
  nLin = DimBitmap[nImgIn].cy;
  cprop  = (double)nRho/(RhoMax-RhoMin);

  if (imgBin) {  // Imagem binária
    for (k=0; k<nTheta; k++) {
      Theta = (ThetaMin+k*dTheta)*GRAD;
      for (i=0; i<nLin; i++)
        for (j=0; j<nCol; j++)
          if (PixImg[nImgIn][i*nCol+j] == 255)
            for (l=-w; l<=w; l++) {
              Rho = ((double)j*cos(Theta)+(double)i*sin(Theta)-RhoMin)*cprop+(double)l;
              if ((Rho>=RhoMin) || (Rho<=RhoMax))
                H[k*nRho+(int)Rho]+= 1.0-2.0*l*l/(w*w)+l*l*l*l/(w*w*w*w);
            }
    }
  }
  else {            // Imagem em tons de cinza
    hM = GlobalAlloc(GHND, (DWORD)(nTheta*nRho*sizeof(int)));
    Cont = (int *) GlobalLock(hM);
    for (k=0; k<nTheta; k++) {
      Theta = (ThetaMin+k*dTheta)*GRAD;
      for (i=0; i<nLin; i++)
        for (j=0; j<nCol; j++) {
          Rho = ((double)j*cos(Theta)+(double)i*sin(Theta)-RhoMin)*cprop;
          m = k*nRho+(int)Rho;
          H[m]+= (double)PixImg[nImgIn][i*nCol+j];
          Cont[m]++;
        }
    }

    // Normalização da matriz H

    maxCont = 0;
    sImg = nTheta*nRho;
    for (i=0; i<sImg; i++)
      if (Cont[i]>maxCont)
        maxCont = Cont[i];

    // Liberação de memória

    GlobalUnlock(hM);
  }

  #undef w

  return;
}

/*****************************************************************************************************
 MaxVetor
 --------
 Determina o valor máximo de um vetor.
 Parâmetros:
   n: dimensão do vetor
   X: vetor
 *****************************************************************************************************/
double MaxVetor(int n, double *X)
{
  int i;       // índice
  double Xmax; // valor máximo do vetor

  Xmax = 0.0;
  for (i=0; i<n; i++)
    if (X[i] > Xmax)
      Xmax = X[i];

  return(Xmax);
}

/*****************************************************************************************************
 AchaRetasParalelas
 ------------------
 Determina o número de retas da imagem e suas posições (Rho, Theta) no espaço de Hough, admitindo que as
 retas são paralelas.
 Parâmetros:
   nImg     : imagem com as retas localizadas no espaço de Hough
   H        : matriz de Hough
```

```
   nTheta   : número de linhas de H
   ThetaMin: valor mínimo de Theta
   dTheta   : incremento em Theta
   nRho     : número de colunas de H
   RhoMin   : valor mínimo de Rho
   dRho     : incremento em Rho
   b        : meia largura da vizinhança de interpolação
   nfxPic   : TRUE -> localização de um número fixo de picos
   nPic     : número de picos a serem localizados
   Fcorte   : limiar do espaço de Hough
 ********************************************************************************************************/
void AchaRetasParalelas(int nImg,double *H,int nTheta,double ThetaMin,double dTheta,int nRho,double RhoMin,
                        double dRho,double b,BOOL nfxPic,int nPic,double Fcorte)
{
  HGLOBAL hM1,hM2,hM3,hM4;  // manipuladores de memória
  double *G;                // gradiente do espaço H segundo theta=const (ou seja: dH/drho)
  double *SomaGth;          // vetor com as somatórias de G para cada valor theta = const
  double MaxSomaG;          // máximo valor de SomaGth
  int iMaxSomaG;            // índice de theta correspondente a MaxSomaG
  double iThGmax;           // índice real interpolado correspondente a iMaxSomaG
  double th;                // valor de theta correspondente a iThGmax
  double *Hint;             // vetor com os valores interpolados de H na posição theta = iThGmax
  int iThAnt,iThPos;        // índices theta adjacentes a iThGmax
  double del_i;             // variável auxiliar: posição de iThGmax referida a iThAnt
  double Hmax;              // valor máximo de H ao longo de theta = iThGmax
  double *iRhHmax;          // vetor de índices de coordenadas radiais ao longo de theta = iThGmax
  double MaxLocH;           // máximo local de H
  BOOL ExisteMaxLoc;        // TRUE-> enquanto forem encontrados máximos locais em H
  int i,j,k;                // índices
  int nRetas;               // número de retas localizadas no espaço H

  // Cálculo do gradiente das projeções: G = dH/drho(theta=const)

  hM1 = GlobalAlloc(GHND, (DWORD)(nTheta*nRho*sizeof(double)));
  G = (double*) GlobalLock(hM1);
  for (i=0; i<nTheta; i++)
    for (j=1; j<nRho-1; j++) {
      k= i*nRho+j;
      G[k] = fabs(H[k+1]-H[k-1]);  // Operador adotado: [-1 0 +1]
    }

  // Cálculo das somatórias de G ao longo de theta=const

  hM2 = GlobalAlloc(GHND, (DWORD)(nTheta*sizeof(double)));
  SomaGth = (double*) GlobalLock(hM2);
  MaxSomaG = 0.0;
  for (i=0; i<nTheta; i++) {
    for (j=1; j<nRho-1; j++)
      SomaGth[i]+= G[i*nRho+j];
    if (SomaGth[i] > MaxSomaG) {
      MaxSomaG  = SomaGth[i];
      iMaxSomaG = i;
    }
  }

  // Localização do ângulo theta (interpolado) para o qual a somatória de gradientes é máxima

  iThGmax = CalcCentroide(SomaGth, nTheta, iMaxSomaG);

  // Interpolação linear de H ao longo de theta = iThGmax

  hM3 = GlobalAlloc(GHND, (DWORD)(nRho*sizeof(double)));
  Hint = (double*) GlobalLock(hM3);
  iThAnt = (int)floor(iThGmax);
  iThPos = (int) ceil(iThGmax);
  del_i  = (double)(iThGmax)-iThAnt;
  Hmax = 0.0;
  for (j=0; j<nRho; j++) {
    Hint[j] = H[iThAnt*nRho+j]+del_i*(H[iThPos*nRho+j]-H[iThAnt*nRho+j]);
    if (Hint[j]>Hmax)
      Hmax = Hint[j];
  }

  // Determinação dos máximos locais de H ao longo de theta = iThGmax

  hM4 = GlobalAlloc(GHND, (DWORD)(nRho*sizeof(double)));
  iRhHmax = (double*) GlobalLock(hM4);
  nRetas = 0;
  if (nfxPic)
    for (i=0; i<nPic; i++) {
      AchaMaxLoc1D(Hint, nRho, (int)b, &iRhHmax[nRetas]);
      nRetas++;
    }
  else {
```

```
    MaxLocH = Hmax;
    ExisteMaxLoc = TRUE;
    while (ExisteMaxLoc && (nRetas<NMAX_RETAS_IMG)) {
      MaxLocH = AchaMaxLoc1D(Hint, nRho, (int)b, &iRhHmax[nRetas]);
      if (MaxLocH > Fcorte*Hmax)
        nRetas++;
      else
        ExisteMaxLoc = FALSE;
    }
  }

  // Determinação do ângulo das retas e de suas posições radiais

  th = ThetaMin+iThGmax*dTheta;
  for (i=0; i<nRetas; i++) {
    FeixeRetas[nImg].Angulo[i] = th;
    FeixeRetas[nImg].Raio  [i] = RhoMin+iRhHmax[i]*dRho;
  }
  FeixeRetas[nImg].nRetasImg = nRetas;

  // Liberação de memória

  GlobalUnlock(hM4);
  GlobalUnlock(hM3);
  GlobalUnlock(hM2);
  GlobalUnlock(hM1);

  return;
}

/*****************************************************************************************************
 AchaRetas
 ---------
 Determina o número de retas da imagem e suas posições (Rho, Theta) no espaço de Hough.
 Parâmetros:
   nImg     : imagem com as retas localizadas no espaço de Hough
   H        : matriz de Hough
   nTheta   : número de linhas de H
   ThetaMin: valor mínimo de Theta
   dTheta   : incremento em Theta
   nRho     : número de colunas de H
   RhoMin   : valor mínimo de Rho
   dRho     : incremento em Rho
   b        : meia largura da vizinhança de interpolação
   nfxPic   : TRUE -> localização de um número fixo de picos
   nPic     : número de picos a serem localizados
   Fcorte   : limiar do espaço de Hough
 *****************************************************************************************************/
void AchaRetas(int nImg,double *H,int nTheta,double ThetaMin,double dTheta,int nRho,double RhoMin,
               double dRho,double b,BOOL nfxPic,int nPic,double Fcorte)
{
  HGLOBAL hM1,hM2;           // manipuladores de memória
  double Hmax;               // valor máximo global de H
  double *iThHmax;           // vetor dos índices da coordenada angular que maximizam H
  double *iRhHmax;           // vetor dos índices da coordenadaa radial que maximizam H
  double MaxLocH;            // máximo local de H
  BOOL ExisteMaxLoc;         // TRUE-> enquanto forem encontrados máximos locais em H
  int i;                     // índice
  int nRetas;                // número de retas localizadas no espaço H

  // Criação dos vetores de índices associados aos máximos locais de H

  hM1 = GlobalAlloc(GHND, (DWORD)(nTheta*nRho*sizeof(double)));
  iThHmax = (double*) GlobalLock(hM1);
  hM2 = GlobalAlloc(GHND, (DWORD)(nTheta*nRho*sizeof(double)));
  iRhHmax = (double*) GlobalLock(hM2);

  // Localizacão das retas no espaço de Hough

  nRetas = 0;
  if (nfxPic)   // Número fixo de retas
    for (i=0; i<nPic; i++) {
      AchaMaxLoc2D(H, nRho, nTheta, (int)b, &iRhHmax[nRetas], &iThHmax[nRetas]);
      nRetas++;
    }
  else {        // Retas correspondentes a máximos locais maiores que um certo limiar
    Hmax = MaxVetor(nTheta*nRho, H);
    MaxLocH = Hmax;
    ExisteMaxLoc = TRUE;
    while (ExisteMaxLoc && (nRetas<NMAX_RETAS_IMG)) {
      MaxLocH = AchaMaxLoc2D(H, nRho, nTheta, (int)b, &iRhHmax[nRetas], &iThHmax[nRetas]);
      if (MaxLocH > Fcorte*Hmax)
        nRetas++;
      else
```

```
        ExisteMaxLoc = FALSE;
      }
    }

    // Determinação do ângulo das retas e de suas posições radiais

    for (i=0; i<nRetas; i++) {
      FeixeRetas[nImg].Angulo[i] = ThetaMin+iThHmax[i]*dTheta;
      FeixeRetas[nImg].Raio  [i] = RhoMin+iRhHmax[i]*dRho;
    }
    FeixeRetas[nImg].nRetasImg = nRetas;

    // Liberação de memória

    GlobalUnlock(hM2);
    GlobalUnlock(hM1);

    return;
}

/*****************************************************************************************************
 AchaMaxLoc1D
 ------------
 Algoritmo para localização de máximos locais de um vetor y.
 Parâmetros:
   y: vetor de números reais
   n: número de elementos do vetor y
   b: meia-largura da vizinhança ao redor do máximo de y
   x: posição do máximo valor de y
 *****************************************************************************************************/
double AchaMaxLoc1D(double *y, int n, int b, double *x)
{
  int i,i0,i1;     // índices
  double ymax;     // valor máximo do vetor y
  int imax;        // índice do vetor y correspondente a ymax

  // Localiza o valor máximo do vetor y

  ymax = 0;
  for (i=0; i< n; i++)
    if (y[i]>ymax) {
      ymax = y[i];
      imax = i;
    }

  // Utiliza as vizinhanças de imax para obter o valor interpolado do máximo local de y

  *x = CalcCentroide(y, n, imax);

  // Anula em y os elementos da vizinhança [imax-b, imax+b]

  i0 = b;
  i1 = b;
  if (imax < b)
    i0 = imax;
  else if (imax > n-b-1)
    i1 = n-1-imax;
  for (i=imax-i0; i<=imax+i1; i++)
    y[i] = 0;

  return (ymax);
}

/*****************************************************************************************************
 AchaMaxLoc2D
 ------------
 Algoritmo para localização de máximos locais de uma matriz Z.
 Parâmetros:
   Z : matriz de números reais
   nx: número de colunas de Z
   ny: número de linhas de Z
   b : meia-largura da vizinhança ao redor do máximo de Z
   x : coluna correspondente ao máximo valor de Z
   y : linha correspondente ao máximo valor de Z
 *****************************************************************************************************/
double AchaMaxLoc2D(double *Z, int nx, int ny, int b, double *x, double *y)
{
  int i,j,k;            // índices
  double Zmax;          // valor máximo da matriz Z
  int xZmax, yZmax;     // índices dos máximos locais de Z

  // Localiza o valor máximo da matriz Z

  Zmax = 0;
```

```
  for (i=0; i<ny; i++)
    for (j=0; j<nx; j++) {
      k = i*nx+j;
      if (Z[k]>Zmax) {
        Zmax = Z[k];
        xZmax = j;
        yZmax = i;
      }
    }

  // Anula em Z os elementos da vizinhança
  // V(x,y) | x está em [xZmax-b, xZmax+b] e y está em [yZmax-b,yZmax+b]

  int i0 = yZmax-b;
  int i1 = yZmax+b;
  int j0 = xZmax-b;
  int j1 = xZmax+b;

  for (i=i0; i<=i1; i++)
    for (j=j0; j<=j1; j++)
      if ((i>=0) && (i<ny) && (j>=0) && (j<nx)) {
        k = i*nx+j;
        Z[k] = 0;
      }
  *x = (double)xZmax;
  *y = (double)yZmax;

  return (Zmax);
}

/*****************************************************************************************************************
 CalcCentroide
 -------------
 Determina o centróide da distribuição de um vetor em torno de seu valor máximo utilizando uma janela de
 raio unitário. O centróide é calculado através de momentos de ordem 0 e 1.
 Parâmetros:
   X    : vetor
   n    . tamanho do vetor
   iMax: índice correspondente ao valor máximo
 Retorna:
   posição do centróide da distribuição em torno de iMax
 *****************************************************************************************************************/
double CalcCentroide(double *X, int n, int iMax)
{
  int i;        // índice
  double xg;    // centróide da distribuição de  X em torno de iMax
  double M0,M1; // momentos de ordem zero e 1

  if(iMax==0)
    xg = X[1]/(X[0]+X[1]);
  else if (iMax==n-1)
    xg = (X[n-2]*(n-2)+X[n-1]*(n-1))/(X[n-2]+X[n-1]);
  else {
    M0 = 0.0;
    M1 = 0.0;
    for (i=-1; i<=1; i++) {
      M0+= X[iMax+i];
      M1+= (iMax+i)*X[iMax+i];
    }
    xg = M1/M0;
  }

  return(xg);
}

/*****************************************************************************************************************
 AjustaRetas1
 ------------
 Aplica à imagem nImgBor máscaras localizadoras das nuvens de pixels correspondentes a cada uma das retas
 da imagem nImgRtAp. Em seguida, utiliza o método dos mínimos quadrados para ajustar as nuvens de pixels a
 um feixe de retas paralelas.
 Parâmetros:
   nImgBor : imagem de bordas
   nImgRtAp: imagem das retas aproximadas
   nImgRtAj: imagem das retas ajustadas
   largJan : largura da janela de busca
   Dup     : TRUE-> imagem de dupla resolução
 *****************************************************************************************************************/
void AjustaRetas1(int nImgBor, int nImgRtAp, int nImgRtAj, double largJan, BOOL Dup)
{
  int i,j;                          // índices
  int nRetas;                       // número de retas da imagem nImgRtAp
  double raio, angulo;              // coordenadas da reta aproximada
  float **X;                        // abcissas X dos pixels em uma faixa da imagem de bordas
```

```
float **Y;                              // ordenadas Y dos pixels em uma faixa da imagem de bordas
HGLOBAL hM1, hM2, hM3;                  // manipuladores de memória
double A, B;                            // parâmetros da equação do erro quadrático
double *C;                              // vetor de parâmetros da equação do erro quadrático
double *D;                              // vetor de parâmetros da equação do erro quadrático
int *n;                                 // número de pontos de uma dada reta do feixe
double Alfa,Beta,Delta,Gama,Lambda;     // parâmetros da equação do erro quadrático mínimo
double angm;                            // ângulo médio formado pelo feixe de retas
double f;                               // função derivada do erro quadrático em relação ao ângulo
double fMin;                            // valor mínimo de f
double angMin;                          // valor de ângulo que minimiza f
double d;                               // distância de um pixel de borda a uma reta dada
double sigma;                           // variância dos pixels de bordas em torno de uma dada reta
double cc,ss;                           // seno e cosseno de um dado ângulo

#define VarrAng 0.50                    // varredura angular em torno do valor angm
#define DelAng 0.01                     // precisão utilizada para localizar o ângulo que minimiza f

// Alocação de memória

nRetas = FeixeRetas[nImgRtAp].nRetasImg;
hM1 = GlobalAlloc(GHND, nRetas*sizeof(int));
n = (int*) GlobalLock(hM1);
hM2 = GlobalAlloc(GHND, nRetas*sizeof(double));
C = (double*) GlobalLock(hM2);
hM3 = GlobalAlloc(GHND, nRetas*sizeof(double));
D = (double*) GlobalLock(hM3);
X = matrix(0, nRetas-1, 0, N_PixelsX*8);
Y = matrix(0, nRetas-1, 0, N_PixelsY*8);

// Ajuste das nuvens de pixels a um feixe de retas paralelas

Alfa   = 0;
Beta   = 0;
Gama   = 0;
Delta  = 0;
Lambda = 0;
angm   = 0;
for (i=0; i<nRetas; i++)
  angm+= FeixeRetas[nImgRtAp].Angulo[i];
angm/= (double)nRetas;

// Determinação das coordenadas dos pixels da imagem de bordas em torno das retas aproximadas

for (i=0; i<nRetas; i++) {

  raio = FeixeRetas[nImgRtAp].Raio[i];
  angulo = FeixeRetas[nImgRtAp].Angulo[i];
  n[i] = SelecionaPontos(nImgBor, Dup, largJan, raio, angulo, X[i], Y[i]);
  A = 0;
  B = 0;
  C[i] = 0;
  D[i] = 0;

  // Determinação dos parâmetros das equações do erro quadrático e de suas derivadas primeiras

  for (j=0; j<n[i]; j++) {
    A+= Y[i][j]*Y[i][j]-X[i][j]*X[i][j];
    B+= X[i][j]*Y[i][j];
    C[i]+= X[i][j];
    D[i]+= Y[i][j];
  }
  Alfa  += A;
  Beta  += B;
  Gama  += C[i]*C[i]/(double)n[i];
  Delta += D[i]*D[i]/(double)n[i];
  Lambda+= C[i]*D[i]/(double)n[i];

}

// Obtenção do ângulo angMin que minimiza o erro quadrático atrevés
// de busca exaustiva no intervalo [angm-VarrAng, angm+VarrAng]

angulo = angm-VarrAng;
fMin = 1.0e6;
while (angulo<=angm+VarrAng) {
  f = fabs(0.5*(Alfa+Gama-Delta)*sin(2.0*angulo*GRAD)+(Beta-Lambda)*cos(2.0*angulo*GRAD));
  if (f<fMin) {
    fMin = f;
    angMin = angulo;
  }
  angulo+= DelAng;
}
```

```
  // Obtenção dos raios que minimizam o erro quadrático

  cc = cos(angMin*GRAD);
  ss = sin(angMin*GRAD);
  FeixeRetas[nImgRtAj].nRetasImg = nRetas;
  for (i=0; i<nRetas; i++) {
    FeixeRetas[nImgRtAj].Angulo[i] = angMin;
    FeixeRetas[nImgRtAj].Raio[i] = fabs(C[i]*cc+D[i]*ss)/(double)n[i];
    FeixeRetas[nImgRtAj].nPts[i] = n[i];
  }

  // Determinação do desvio-padrão da reta-média passando pela nuvem de pixels

  for (i=0; i<nRetas; i++) {
    sigma = 0;
    for (j=0; j<n[i]; j++) {
      d = fabs(X[i][j]*cc+Y[i][j]*ss)-FeixeRetas[nImgRtAj].Raio[i]; // --> modificação
      sigma+= d*d;
    }
    sigma = sigma/(double)(n[i]-1);
    FeixeRetas[nImgRtAj].Sigma[i] = sqrt(sigma/(double)n[i]);
  }

  // Apresentação da tabela com os dados do feixe de retas e respectiva imagem

  InfoCaractImg(nImgRtAj,1,42);
  sprintf(NomeImg[nImgRtAj], "Retas Ajustadas");
  ImagemdeRetas(nImgRtAp, nImgRtAj, nRetas, FeixeRetas[nImgRtAj].Angulo, FeixeRetas[nImgRtAj].Raio);


  #undef VarrAng
  #undef DelAng

  // Liberação de memória

  GlobalUnlock(hM1);
  GlobalUnlock(hM2);
  GlobalUnlock(hM3);

  return;
}

/*****************************************************************************************************
 AjustaRetas2
 ------------
 Aplica à imagem nImgBor máscaras localizadoras das nuvens de pixels correspondentes a cada uma das retas
 da imagem nImgRtAp. Em seguida, utiliza o método dos mínimos quadrados para ajustar as nuvens de pixels a
 um feixe de retas quaisquer.
 Parâmetros:
   nImgBor : imagem de bordas
   nImgRtAp: imagem das retas aproximadas
   nImgRtAj: imagem das retas ajustadas
   largJan : largura da janela de busca
   Dup     : TRUE-> imagem de dupla resolução
 *****************************************************************************************************/
void AjustaRetas2(int nImgBor, int nImgRtAp, int nImgRtAj, double largJan, BOOL Dup)
{
  int i,j;                                  // índices
  int nRetas;                               // número de retas da imagem nImgRtAp
  double raioAp, anguloAp;                  // descritores da reta aproximada
  float X[N_PixelsX*8];                     // abcissas X dos pixels em uma faixa da imagem de bordas
  float Y[N_PixelsX*8];                     // ordenadas Y dos pixels em uma faixa da imagem de bordas
  double C,D,Alfa,Beta,Delta,Gama,Lambda;   // parâmetros da equação do erro quadrático
  int n;                                    // número de pontos de uma dada reta do feixe
  double f;                                 // função derivada do erro quadrático em relação ao ângulo
  double fMin;                              // valor mínimo de f
  double angMin;                            // valor do ângulo que minimiza f
  double angulo;                            // ângulo que testa valor mínimo de f
  double d;                                 // distância de um pixel de borda a uma reta dada
  double sigma;                             // variância dos pixels de bordas em torno de uma dada reta
  double cc,ss;                             // cosseno e seno de um dado ângulo

  #define VarrAng 5.0                       // varredura angular em torno do valor angm
  #define DelAng 0.01                       // precisão utilizada para localizar o ângulo que minimiza f

  // Ajuste das nuvens de pontos

  if (largJan>0)
    nRetas = FeixeRetas[nImgRtAp].nRetasImg;
  else
    nRetas = -(int)largJan;

  for (i=0; i<nRetas; i++) {
```

```
    // Determinação das coordenadas dos pixels da imagem de bordas em torno das retas aproximadas

    anguloAp = FeixeRetas[nImgRtAp].Angulo[i];
    if (largJan>0) {
      raioAp = FeixeRetas[nImgRtAp].Raio[i];
      n = SelecionaPontos(nImgBor, Dup, largJan, raioAp, anguloAp, X, Y);
    }
    else
      n = SelecionaPontos2(nImgBor, nImgRtAp, 255-nRetas+i, X, Y);

    // Determinação dos parâmetros das equações do erro quadrático e de suas derivadas primeiras

    Alfa = 0;
    Beta = 0;
    Gama = 0;
    Delta = 0;
    C = 0;
    D = 0;
    for (j=0; j<n; j++) {
      Alfa+= Y[j]*Y[j]-X[j]*X[j];
      Beta+= X[j]*Y[j];
      C+= X[j];
      D+= Y[j];
    }
    Gama  = C*C/(double)n;
    Delta = D*D/(double)n;
    Lambda= C*D/(double)n;

    // Obtenção do ângulo angMin que minimiza o erro quadrático através
    // de busca exaustiva no intervalo [anguloAp-VarrAng,anguloAp+VarrAng]

    fMin = (Alfa+Gama+Delta)*sin(2*anguloAp*GRAD)-Beta*cos(2*anguloAp*GRAD)+Lambda;
    fMin = fabs(fMin);
    angMin = anguloAp;
    fMin = 1.0e6;
    angulo = anguloAp-VarrAng;
    while (angulo<=anguloAp+VarrAng) {
      f = fabs(0.5*(Alfa+Gama-Delta)*sin(2*angulo*GRAD)+(Beta-Lambda)*cos(2*angulo*GRAD));
      if (f<fMin) {
        fMin = f;
        angMin = angulo;
      }
      angulo+= DelAng;
    }
    FeixeRetas[nImgRtAj].Angulo[i] = angMin;

    // Obtenção do raio que minimiza o erro quadrático

    FeixeRetas[nImgRtAj].Raio[i] =  fabs(C*cos(angMin*GRAD)+D*sin(angMin*GRAD))/(double)n;

    // Determinação do desvio-padrão da reta-média passando pela nuvem de pixels

    sigma = 0;
    cc = cos(angMin*GRAD);
    ss = sin(angMin*GRAD);
    for (j=0; j<n; j++) {
      d = X[j]*cc+Y[j]*ss-FeixeRetas[nImgRtAj].Raio[i];
      sigma+= d*d;
    }
    sigma = sigma/(double)(n-1);
    FeixeRetas[nImgRtAj].Sigma[i] = sqrt(sigma/(double)n);
    FeixeRetas[nImgRtAj].nPts [i] = n;
  }

  // Apresentação da tabela com os dados do feixe de retas e respectiva imagem

  FeixeRetas[nImgRtAj].nRetasImg = nRetas;
  InfoCaractImg(nImgRtAj,1,42);
  sprintf(NomeImg[nImgRtAj], "Img%d: Retas Ajustadas", nImgRtAj);
  ImagemdeRetas(nImgRtAp, nImgRtAj, nRetas, FeixeRetas[nImgRtAj].Angulo, FeixeRetas[nImgRtAj].Raio);


  #undef VarrAng
  #undef DelAng

  return;
}

/*****************************************************************************************************
 SelecionaPontos
 ---------------
 Seleciona os pixels da imagem nImgBor distando d < delta da reta R(rho,theta).
 Parâmetros:
   nImgBor: imagem de bordas
```

```
   Dup    : TRUE-> imagem em dupla resolução
   delta  : largura da janela de busca
   rho    : distância da reta à origem
   theta  : ângulo da reta com a vertical
   X      : abcissas dos pixels localizados
   Y      : ordenadas dos pixels localizados
 ***************************************************************************************************************/
int SelecionaPontos(int nImgBor, BOOL Dup, double delta, double rho, double theta, float *X, float *Y)
{
  int nLin, nCol;       // número de linhas e de colunas da imagem
  int i,j,k;            // índices
  int m;                // número de pontos encontrados
  float x, y;           // coordenadas de um ponto
  double d;             // distância entre um ponto e a reta (rho,theta)
  double c,s;           // cosseno e seno do ângulo theta
  int xMax,yMax;        // valores máximos admissíveis para as coordenadas x,y

  // Recuperação de informações básicas das imagens

  if (Dup) {
    nCol = 2*DimBitmap[nImgBor].cx;
    nLin = 2*DimBitmap[nImgBor].cy;
  }
  else {
    nCol = DimBitmap[nImgBor].cx;
    nLin = DimBitmap[nImgBor].cy;
  }
  xMax = DimBitmap[nImgBor].cx;
  yMax = DimBitmap[nImgBor].cy;

  // Seleção dos pixels situados a distâncias r<delta da reta referência

  m = 0;
  c = cos(theta*GRAD);
  s = sin(theta*GRAD);
  for (i=0; i<nLin; i++)
    for (j=0; j<nCol; j++) {
      k = i*nCol+j;
      x = CoordBordas.XBorda[k];
      y = CoordBordas.YBorda[k];
      if ((x>=0) && (x<xMax) && (y>=0) && (y<yMax)) {
        d = fabs(fabs(x*c+y*s)-fabs(rho));
        if (d<=delta) {
          X[m] = x;
          Y[m] = y;
          m++;
        }
      }
    }

  return(m);
}

/***************************************************************************************************************
 SelecionaPontos2
 ----------------
 Seleciona os pixels da imagem nImg1 pertencentes à máscara nmask da imagem nImg2
 Parâmetros:
   nImg1: imagem segmentada
   nImg2: imagem de máscaras rotuladas
   nmask: índice da máscara
   X    : abcissas dos pixels localizados
   Y    : ordenadas dos pixels localizados
 ***************************************************************************************************************/
int SelecionaPontos2(int nImg1, int nImg2, int nmask, float *X, float *Y)
{
  int sImg; // tamanho da imagem
  int i;    // índices
  int m;    // número de pixels encontrados

  sImg = DimBitmap[nImg1].cx*DimBitmap[nImg1].cy;
  m = 0;
  for (i=0; i<sImg; i++)
    if ((PixImg[nImg1][i]>0) && (PixImg[nImg2][i]==(BYTE)nmask)) {
      X[m] = CoordBordas.XBorda[i];
      Y[m] = CoordBordas.YBorda[i];
      m++;
    }

  return(m);
}

/***************************************************************************************************************
 IdFreqESPRIT
```

```
------------
Identifica as frequências espaciais significativas da imagem utilizando o algoritmo TLS-ESPRIT
("Estimation of Signal Parameters via Rotational Invariance Techniques").
Parâmetros:
  nImgIn: imagem original
  nF    : número de frequências espaciais
  Vert  : TRUE->período fundamental vertical
*****************************************************************************************************************/
double IdFreqESPRIT(int nImgIn, int nF, BOOL Vert)
{
  int i,j,k;            // índices
  int nLin, nCol;       // número de linhas e de colunas da imagem original
  int nFreq;            // número de frequências
  double *x;            // vetor de medidas: x[1..nLin]
  double **X;           // matriz de medidas: X[1..m][1..nLin-m+1]
  int m;                // número de linhas da matriz [X]
  double **Rxx;         // matriz de covariância de [X]: Rxx[1..m][1..m]
  int nRot;             // no. de rotações-Jacobi para geração dos auto-valores
  double *Lambda;       // vetor de auto-valores: Lambda[1..m] ou Lambda[1..2m-2] conforme o caso
  double **E;           // matriz de autovetores: E[1..m][1..m] ou E[1..2m-2][1..2m-2] conforme o caso
  double **G;           // matriz englobando 2 submatrizes E1[1..m-1][1..nFreq] e E2[2..m][1..nFreq]
  double **H;           // matriz produto de [G] por sua transposta: [H] = [G].[Gt]; H[0..2m-2][0..nFreq]
  double **P;           // sub-matriz [H12] de [H]: P[0..m-1][0..m-1]
  double **Q;           // sub-matriz H[22] de [H]: Q[0..m-1][0..m-1]
  double **Qori;        // matriz cópia de Q
  double **R;           // matriz inversa de [Q]: R[0..m-1][0..m-1]
  int *indx;            // vetor com o registro das permutações de linhas efetuadas pela função ludcmp
  double *col;          // vetor auxiliar, usado na rotina de inversão de matrizes
  double *colOri;       // vetor-cópia de col
  double d;             // variável não-utilizada, devolvida pela função ludcmp
  double **S;           // matriz [S] = -[P].[R] ; S[0..m-1][0..m-1]
  double *Wr, *Wi;      // vetores com as partes real [Wr] e imaginária [Wi] dos autovalores de [S]
  double T;             // período da imagem

  // Recuperação de informaçoes da imagem original

  nCol = DimBitmap[nImgIn].cx;
  nLin = DimBitmap[nImgIn].cy;
  nFreq = nF*2;

  // 1. Construção do vetor de medidas: x[1..nLin] ou x[i..nCol] conforme a direção escolhida

  if (Vert) { // cálculo do período fundamental na direção vertical
    m = (int)sqrt(nLin);
    x = dvector(1, nLin);
    double xm=0;
    for (i=0; i<nLin; i++) {
      k = i*nCol;
      for (j=0; j<nCol; j++)
        x[i+1]+= PixImg[nImgIn][k+j];
      xm+= x[i+1];
    }
    xm/= nLin;
    for (i=1; i<=nLin; i++)
      x[i]-= xm;
    X = dmatrix(1, m, 1, nLin-m+1);
  }
  else {      // cálculo do período fundamental na direção horizontal
    m = (int)sqrt(nCol);
    x = dvector(1, nCol);
    double xm=0;
    for (j=0; j<nCol; j++) {
      for (i=0; i<nLin; i++) {
        k = i*nCol+j;
        x[j+1]+= PixImg[nImgIn][k];
      }
      xm+= x[j+1];
    }
    xm/= nCol;
    for (i=1; i<nCol; i++)
      x[i]-= xm;
    X = dmatrix(1, m, 1, nCol-m+1);
  }

  // 2. Construção da matriz X de medidas: X[1..m][1..nLin-m+1]

  k = 0;
  for (j=1; j<=nLin-m+1; j++) {
    for (i=1; i<=m; i++) {
      X[i][j] = x[i+k];
    }
    k++;
  }
```

```
// 3. Construção da matriz [Rxx] de covariância de [X]: Rxx[1..m][1..m]

Rxx = dmatrix(1, m, 1, m);
for (i=1; i<=m; i++) {
  for (j=1; j<=m; j++) {
    Rxx[i][j] = 0;
    for (k=1; k<=nLin-m+1; k++)
      Rxx[i][j]+= X[i][k]*X[j][k];
    Rxx[i][j]/= (double)(nLin-m+1);
  }
}

// 4. Determinação dos autovalores [Lambda] e autovetores [E] de [Rxx]: Lambda[1..m]; E[1..m][1..m]

Lambda = dvector(1, m);
E = dmatrix(1, m, 1, m);
jacobi(Rxx, m, Lambda, E, &nRot);
eigsrt(Lambda, E, m);

//
//                  |[E1*]|             |e(1,1)   ... e(1,nFreq)  |             |e(2,1)     e(2,nFreq) |
// 6. Matriz [G]: G=|     | onde [E1] = |                        |   e [E2] = |                        |
//                  |[E2*]|             |e(m-1,1) ... e(m-1,nFreq)|             |e(m,1) ... e(m,nFreq) |
//

G = dmatrix(1, 2*nFreq, 1, m);
for (i=1; i<=nFreq; i++)
  for (j=1; j<=m-1; j++) {
    G[i][j] = E[j][i];
    G[i+nFreq][j] = E[j+1][i];
  }

// 7. Matriz [H]: [G][G*]   :   H[1..2nFreq][1..2nFreq]

H = dmatrix(1, 2*nFreq, 1, 2*nFreq);
for (i=1; i<=2*nFreq; i++)
  for (j=1; j<=2*nFreq; j++) {
    H[i][j] = 0;
    for (k=1; k<=nFreq; k++)
      H[i][j]+= G[i][k]*G[j][k];
  }

// Cálculo dos autovetores [Lambda] e autovetores [E] de [H]

Lambda = dvector(1, 2*nFreq);
E = dmatrix(1, 2*nFreq, 1, 2*nFreq);
jacobi(H, 2*nFreq, Lambda, E, &nRot);
eigsrt(Lambda, E, 2*nFreq);

//                              |[H11] [H12] |
// 8. Matrizes P e Q => seja H=|            |
//                              |[H21] [H22] |
//
// onde P = [H12]: P[1..nFreq][1..nFreq],  Q = [H22]: Q[1..nFreq][1..nFreq]

P = dmatrix(1, nFreq, 1, nFreq);
Q = dmatrix(1, nFreq, 1, nFreq);

for (i=1; i<=nFreq; i++)
  for (j=1; j<=nFreq; j++) {
    P[i][j] = E[i][j+nFreq];
    Q[i][j] = E[i+nFreq][j+nFreq];
  }

// Matriz R = inv(Q)

R = dmatrix(1, nFreq, 1, nFreq);

// Matrizes auxiliares [indx] e [col]: indx[0..nFreq] e col[0..nFreq]

indx = ivector(1, nFreq);
col = dvector(1, nFreq);

// Inversão da matriz [Q]: [R] = Inv([Q])

colOri = dvector(1,nFreq);
Qori = dmatrix(1,nFreq,1,nFreq);
for (i=1; i<=nFreq; i++)
  for (j=1; j<=nFreq; j++)
    Qori[i][j] = Q[i][j];
ludcmp(Q, nFreq, indx, &d);
for (j=1; j<=nFreq; j++) {
  for (i=1; i<=nFreq; i++) {
    col[i] = 0.0;
```

```
      colOri[i] = 0.0;
    }
    col[j] = 1.0;
    colOri[j] = 1.0;
    lubksb(Q, nFreq, indx, col);
    mprove(Qori,Q,nFreq,indx,colOri,col);
    for (i=1; i<=nFreq; i++)
      R[i][j] = col[i];
  }

  // Determinação da matriz produto [S] = -[P]*[R]

  S = dmatrix(1, nFreq, 1, nFreq);
  for (i=1; i<=nFreq; i++)
    for(j=1; j<=nFreq; j++) {
      S[i][j] = 0.0;
      for (k=1; k<=nFreq; k++)
        S[i][j]+= -P[i][k]*R[k][j];
    }

  // Cálculo dos autovalores da matriz [S] (real, não-simétrica)

  balanc(S, nFreq);
  elmhes(S, nFreq);
  for (j=1; j<=nFreq-2;j++)
    for (i=j+2; i<=nFreq; i++)
      S[i][j] = 0;
  Wr = dvector(1, nFreq);
  Wi = dvector(1, nFreq);
  hqr(S, nFreq, Wr, Wi);

  // Cálculo do período espacial fundamental da imagem na direção selecionada

  if (Wi[1]!=0.0)
    T = fabs(2*PI/atan(Wi[1]/Wr[1]));

  return T;
}

/*****************************************************************************************************
 CriaElemEstCirc
 ---------------
 Cria um elemento estruturante Círculo com raio dado.
 Parâmetros:
   stdim1 : raio do círculo
   strelem: elemento estruturante
 Retorna:
   número de pontos do elemento estruturante
 *****************************************************************************************************/
int CriaElemEstCirc(int stdim1, ponto *strelem)
{
  int i, j, k;      // indices

  k=0;
  for (i=-stdim1; i<=stdim1; i++)
    for (j=-stdim1; j<=stdim1; j++)
      if (sqrt((double)(i*i+j*j)) <= (double)stdim1) {
        strelem[k].x = j;
        strelem[k].y = i;
        k++;
      }
  return (k);
}

/*****************************************************************************************************
 CriaElemEstAro
 --------------
 Cria um elemento estruturante "coroa circular" com raio dado e espessura unitária.
 Parâmetros:
   raio    : raio externo da coroa circular
   strelem: elemento estruturante
 Retorna:
   número de pontos do elemento estruturante
 *****************************************************************************************************/
int CriaElemEstAro(int raio, ponto *strelem)
{
  int i, j, k;      // indices

  k=0;
  for (i=-raio; i<=raio; i++)
    for (j=-raio; j<=raio; j++)
      if ((sqrt((double)(i*i+j*j)) <= (double)raio) &&
          (sqrt((double)((i-1)*(i-1)+(j-1)*(j-1))) >= (double)(raio-1))) {
        strelem[k].x = j;
```

```
        strelem[k].y = i;
        k++;
      }
  return (k);
}

/*****************************************************************************************************
 CriaElemEstRet
 --------------
 Cria um elemento estruturante "Retângulo" com largura e altura dadas.
 Parâmetros:
   a: meia-largura do retângulo
   b: meia-altura do retângulo
 Retorna:
   número de pontos do elemento estruturante
 *****************************************************************************************************/
int CriaElemEstRet(float a, float b, ponto *strelem)
{
  int i, j, k;     // indices

  int ia = (int)a;
  int ib = (int)b;
  k=0;
  for (i=-ia; i<=ia; i++)
    for (j=-ib; j<=ib; j++) {
      strelem[k].x = j;
      strelem[k].y = i;
      k++;
    }

  return (k);
}

/*****************************************************************************************************
 CriaElemEstAroRet
 -----------------
 Cria um elemento estruturante "Coroa Retangular" com largura e largura e altura dadas e espessura unitária.
 Parâmetros:
   a       : meia-largura externa da coroa retangular
   b       : meia-altura externa da coroa retangular
   strelem: elemento estruturante
 Retorna:
   número de pontos do elemento estruturante
 *****************************************************************************************************/
int CriaElemEstAroRet(float a, float b, ponto *strelem)
{
  int i, j, k;     // indices

  int ia = (int)a;
  int ib = (int)b;
  k=0;
  for (i=-ia; i<=ia; i++)
    for (j=-ib; j<=ib; j++)
      if ((j==-ib) || (j==ib) || (i==-ia) || (i==ia)) {
        strelem[k].x = j;
        strelem[k].y = i;
        k++;
      }

  return (k);
}

/*****************************************************************************************************
 CriaElemEst5Pts
 ---------------
 Cria um elemento estruturante composto por 5 pontos assim  distribuídos: 1 ponto central e 4 pontos nas
 posições +r1.n1,-r1.n1,+r2.n2,-r2.n2, onde n1,n2 são versores com orientações ang1 e ang2 relativamente
 ao eixo vertical.
 Parâmetros:
   r1,r2     : distâncias dos pontos à origem
   ang1,ang2: orientações dos versores
   strelem   : elemento estruturante
 *****************************************************************************************************/
void CriaElemEst5Pts(double r1, double ang1, double r2, double ang2, ponto *strelem)
{
  strelem[0].x = 0;
  strelem[0].y = 0;
  strelem[1].x =  (int)(r1*sin(ang1)+0.5);
  strelem[1].y =  (int)(r1*cos(ang1)+0.5);
  strelem[2].x = -(int)(r1*sin(ang1)+0.5);
  strelem[2].y = -(int)(r1*cos(ang1)+0.5);
  strelem[3].x =  (int)(r2*sin(ang2)+0.5);
  strelem[3].y = -(int)(r2*cos(ang2)+0.5);
  strelem[4].x = -(int)(r2*sin(ang2)+0.5);
```

```
  strelem[4].y =  (int)(r2*cos(ang2)+0.5);

  return;
}

/*****************************************************************************************************
 CriaElemEst3Pts
 ---------------
 Cria um elemento estruturante composto por 3 pontos assim  distribuídos: 1 ponto central e 2 pontos nas
 posições +r1.n1,-r1.n1, onde n1 é o versor com orientação ang1 relativamente ao eixo vertical.
 Parâmetros:
   r      : distâncias dos pontos à origem
   ang    : orientação do versor
   strelem: elemento estruturante
 *****************************************************************************************************/
void CriaElemEst3Pts(double r, double ang, ponto *strelem)
{
  strelem[0].x = 0;
  strelem[0].y = 0;
  strelem[1].x = (int)floor(r*cos(ang));
  strelem[1].y =-(int)floor(r*sin(ang));
  strelem[2].x =-(int)floor(r*cos(ang));
  strelem[2].y = (int)floor(r*sin(ang));

  return;
}

/*****************************************************************************************************
 Dilatação
 ---------
 Aplica à imagem nImgIn uma operação de dilatação morfológica utilizando o elemento estruturante strelem,
 com np pontos.
 Parâmetros:
   nImgIn : imagem inicial
   nImgOut: imagem dilatada
   np     : número de pontos do elemento estruturante
   strelem: coordenadas dos pontos do elemento estruturante
 *****************************************************************************************************/
void Dilatacao(int nImgIn, int nImgOut, int np, ponto *strelem)
{
  int i;                // índice
  int sbImg;            // tamanho da imagem compactada
  HGLOBAL hM1,hM2,hM3;  // manipuladores de memória global
  BYTE *bImgIn;         // pixels da imagem binária original
  BYTE *t_bImg;         // pixels da imagem binária transladada
  BYTE *bImgOut;        // pixels da imagem binária dilatada

  // Alocação de memória para as imagens compactadas bImgIn, bImgOut e t_bImg

  sbImg = DimBitmap[nImgIn].cx*DimBitmap[nImgIn].cy/8;
  hM1 = GlobalAlloc(GHND, (DWORD)sbImg);
  bImgIn = (BYTE*) GlobalLock(hM1);
  hM2 = GlobalAlloc(GHND, (DWORD)sbImg);
  bImgOut = (BYTE*) GlobalLock(hM2);
  hM3 = GlobalAlloc(GHND, (DWORD)sbImg);
  t_bImg = (BYTE*) GlobalLock(hM3);

  // Dilatação da imagem original compactada

  compact_buffer(nImgIn, bImgIn);   // compacta imagem nImgIn representando 1 pixel por 1 bit
  for (i=0; i<sbImg; i++)
    bImgOut[i] = 0;
  for (i=0; i<np; i++) {            // translada imagem segundo os vetores do elemento estruturante
    img_translate(nImgIn, bImgIn, strelem[i].x, strelem[i].y, t_bImg);
    img_union(t_bImg, bImgOut, sbImg, bImgOut);
  }
  restore_buffer(nImgOut, bImgOut); // restaura imagem dilata representando 1 pixel por 8 bits

  // Liberação de memória

  GlobalUnlock(hM1);
  GlobalUnlock(hM2);
  GlobalUnlock(hM3);
}

/*****************************************************************************************************
 Erosão
 ------
 Aplica à imagem nImgIn uma operação de erosão morfológica utilizando o elemento estruturante strelem, com
 np pontos.
 Parâmetros:
   nImgIn : imagem inicial
   nImgOut: imagem erodida
   np     : número de pontos do elemento estruturante
```

```
   strelem: coordenadas dos pontos do elemento estruturante
 ***************************************************************************************************************/
void Erosao(int nImgIn, int nImgOut, int np, ponto *strelem)
{
  int i;                // índice
  int sbImg;            // tamanho da imagem compactada
  HGLOBAL hM1,hM2,hM3;  // manipuladores de memória global
  BYTE *bImgIn;         // pixels da imagem binária original
  BYTE *t_bImg;         // pixels da imagem binária transladada
  BYTE *bImgOut;        // pixels da imagem binária erodida

  // Alocação de memória para as imagens compactadas bImgIn, bImgOut e t_bImg

  sbImg = DimBitmap[nImgIn].cx*DimBitmap[nImgIn].cy/8;
  hM1 = GlobalAlloc(GHND, (DWORD)sbImg);
  bImgIn = (BYTE*) GlobalLock(hM1);
  hM2 = GlobalAlloc(GHND, (DWORD)sbImg);
  bImgOut = (BYTE*) GlobalLock(hM2);
  hM3 = GlobalAlloc(GHND, (DWORD)sbImg);
  t_bImg = (BYTE*) GlobalLock(hM3);

  // Transposição do elemento estruturante

  for (i=0; i<np; i++) {
    strelem[i].x = -strelem[i].x;
    strelem[i].y = -strelem[i].y;
  }

  // Erosão da imagem original compactada

  compact_buffer(nImgIn, bImgIn);   // compacta imagem nImgIn representando 1 pixel por 1 bit
  img_translate(nImgIn, bImgIn, strelem[0].x, strelem[0].y, bImgOut);
  for (i=1; i<np; i++) {            // translada imagem segundo os vetores do elemento estruturante
    img_translate (nImgIn, bImgIn, strelem[i].x, strelem[i].y, t_bImg);
    img_and(t_bImg, bImgOut, sbImg, bImgOut);
  }
  restore_buffer(nImgOut, bImgOut); // restaura imagem dilatada representando 1 pixel por 8 bits

  // Liberação de memória

  GlobalUnlock(hM1);
  GlobalUnlock(hM2);
  GlobalUnlock(hM3);
}

/***************************************************************************************************************
 Hit_Miss
 --------
 Aplicação de uma operação morfológica "hit-miss" à imagem binária nImgIn.
 Parâmetros:
   nImgIn  : imagem binária original
   nImgOut : imagem processada
   strelem1: elemento estruturante para o "foreground"
   np1     : número de pontos de strelem1
   strelem2: elemento estruturante para o "background"
   np2     : número de pontos de strelem2
 ***************************************************************************************************************/
void Hit_Miss(int nImgIn, int nImgOut, int np1, ponto *strelem1, int np2, ponto *strelem2)
{
  int i;            // índice
  int sImg;         // tamanho da imagem
  HGLOBAL hM;       // manipulador de memória
  LPBYTE imgaux;    // imagem buffer

  // Criação de uma imagem auxiliar

  sImg = DimBitmap[nImgIn].cx*DimBitmap[nImgIn].cy;
  hM = GlobalAlloc(GHND, (DWORD)sImg);
  imgaux = (LPBYTE) GlobalLock(hM);

  // Erosão do "foreground"

  Erosao(nImgIn, nImgOut, np1, strelem1);
  for (i=0; i<sImg; i++)
    imgaux[i] = PixImg[nImgOut][i];

  // Erosão do "background"

  for (i=0; i<sImg; i++)
    PixImg[nImgOut][i] = (PixImg[nImgIn][i]==0) ? 255 : 0;  // imagem do "background"
  Erosao(nImgOut, nImgOut, np2, strelem2);

  // Intersecção do "foreground" e background" erodidos
```

```
  for (i=0; i<sImg; i++)
    PixImg[nImgOut][i] = __min(imgaux[i],PixImg[nImgOut][i]);

  return;
}

/*****************************************************************************************************
 Abertura
 --------
 Aplica à imagem nImgIn uma operação morfológica Abertura.
 Parâmetros:
   nImgIn : imagem original
   nImgOut: imagem aberta
   np     : número de pontos do elemento estruturante
   strelem: coordenadas do elemento estruturante
 *****************************************************************************************************/
void Abertura(int nImgIn, int nImgOut, int np, ponto *strelem)
{
  Erosao(nImgIn, nImgOut, np, strelem);
  Dilatacao(nImgOut, nImgOut, np, strelem);
  return;
}

/*****************************************************************************************************
 Fechamento
 ----------
 Aplica à imagem nImgIn uma operação morfológica Fechamento.
 Parâmetros:
   nImgIn : imagem original
   nImgOut: imagem aberta
   np     : número de pontos do elemento estruturante
   strelem: coordenadas do elemento estruturante
 *****************************************************************************************************/
void Fechamento(int nImgIn, int nImgOut, int np, ponto *strelem)
{
  Dilatacao(nImgIn, nImgOut, np, strelem);
  Erosao(nImgOut, nImgOut, np, strelem);
  return;
}

/*****************************************************************************************************
 DilatacaoCond
 -------------
 Aplica à imagem nImgIn uma dilatação condicional relativamente à imagem referência nImgRef, usando o
 elemento estruturante strelem.
 Parâmetros:
   nImgIn : imagem original
   nImgRef: imagem referência
   nImgOut: imagem dilatada condicionalmente
   np     : número de pontos do elemento estruturante
   strelem: coordenadas do elemento estruturante
 *****************************************************************************************************/
void DilatacaoCond(int nImgIn, int nImgRef, int nImgOut, int np, ponto *strelem)
{
  int cont;          // conta diferenças entre duas imagens
  BOOL not_ok;       // FALSE: quando as duas últimas imagens forem iguais
  HGLOBAL hM;        // manipulador de memória
  BYTE *imgAnt;      // pixels da última imagem processada
  int sImg;          // tamanho da imagem
  int i, j;          // índices
  #define nTolDif 10

  // Criação da "Ultima imagem processada", utilizada para comparações a cada nova dilatação

  sImg = DimBitmap[nImgIn].cx*DimBitmap[nImgIn].cy;
  hM = GlobalAlloc(GHND, (DWORD)sImg);
  imgAnt = (BYTE *)GlobalLock(hM);
  for (i=0; i<sImg; i++) {
    imgAnt[i] = PixImg[nImgIn][i];
    PixImg[nImgOut][i] = PixImg[nImgIn][i];
  }

  // Determina a intersecção entre a imagem dilatada e a imagem-referência
  // enquanto a imagem corrente e a imagem anterior forem iguais

  not_ok = TRUE;
  j=1;
  while (not_ok) {
    // faz nImgOut = nImgOut + strelem
    Dilatacao(nImgOut, nImgOut, np, strelem);
    // intersecção da imagem dilatada nImgOut e da imagem referência nImgRef
    for (i=0; i<sImg; i++)
      PixImg[nImgOut][i] = __min(PixImg[nImgRef][i],PixImg[nImgOut][i]);
    // Apresentação da imagem dilatada condicionalmente
```

```
    SetBitmapBits(hbm[nImgOut], (DWORD)sImg, (LPSTR)PixImg[nImgOut]);
    sprintf(NomeImg[nImgOut], "Dilatação geodésica: passo %d", j);
    MostraImagem(nImgOut);
    // compara com a imagem anterior
    cont = 0;
    for (i=0; i<sImg; i++)
      if (PixImg[nImgOut][i] != imgAnt[i])
        cont++;
    if (cont > nTolDif) // excesso de diferenças => continua o ciclo de dilatações + intersecções
      for (i=0; i<sImg; i++)
        imgAnt[i] = PixImg[nImgOut][i];
    else                // as imagens são iguais => fim da dilatação condicional
      not_ok = FALSE;
    j++;
  }
  #undef nTolDif

  // Liberação de memória

  GlobalUnlock(hM);

  return;
}

/*****************************************************************************************************
 DestroiFronteiricos
 -------------------
 Elimina todos os objetos fronteiricos à moldura da imagem.
 Parâmetros:
   nImgIn : numero da imagem original
   nImgOut: número da imagem processada
   np     : número de pontos do elemento estruturante
   strelem: elemento estruturante
   m1     : espessura da linha horizontal superior do quadro de exclusão
   m2     : espessura da linha vertical direita do quadro de exclusão
 *****************************************************************************************************/
void DestroiObjFront(int nImgIn, int nImgOut, int np, ponto *strelem, int m1, int m2)
{
  int i, j;          // indices
  int nCol, nLin;    // número de colunas e de linhas da imagem
  int sImg;          // tamanho da imagem
  HGLOBAL hM;        // manipulador de memória
  LPBYTE imgMold;    // pixels da imagem moldura

  // Cria imagem da moldura

  nCol = DimBitmap[nImgIn].cx;
  nLin = DimBitmap[nImgIn].cy;
  sImg = nCol*nLin;
  hM = GlobalAlloc(GHND, (DWORD)sImg);
  imgMold = (LPBYTE) GlobalLock(hM);
  for (i=0; i<nLin; i++)
    for (j=0; j<nCol; j++)
      if ((i<=m1) || (j==0) || (i==nLin-1) || (j>nCol-1-m2))
        imgMold[i*nCol+j] = 255;

  // Cria imagem de marcas: intersecção da imagem original com a moldura

  for (i=0; i<sImg; i++)
    PixImg[nImgOut][i] = __min(PixImg[nImgIn][i],imgMold[i]);

  // Reconstrói os objetos que tocam a moldura, começando pelas marcas

  DilatacaoCond(nImgOut, nImgIn, nImgOut, np, strelem);

  // Elimina os objetos que tocam a moldura

  for (i=0; i<sImg; i++)
    if ((PixImg[nImgIn][i] == 255) && (PixImg[nImgOut][i] == 0))
      PixImg[nImgOut][i] = 255;
    else
      PixImg[nImgOut][i] = 0;

  GlobalUnlock(hM);

  return;
}

/*****************************************************************************************************
 TransfDist
 ----------
 Determina a transformada-distância da imagem nImgIn, isto é, associando a cada pixel de nImgIn um nível
 de cinza correspondente à distância daquele pixel ao "background". A distância é medida utilizando-se a
 métrica: d((x1,y1),(x2,y2)) = min{|x2-x1|,|y2-y1|}.
```

```
 Parâmetros:
   nImgIn : número da imagem original
   nImgOut: imagem da transformada-distância de nImgIn
   np     : número de pontos do elemento estruturante
   strelem: elemento estruturante
 ***************************************************************************************************************/
int TransfDist(int nImgIn, int nImgOut, int np, ponto *strelem)
{
  int i;             // índice
  int cont;          // número de pixels "brancos" da imagem
  int maxD;          // máxima distância
  int sImg;          // tamanho da imagem
  HGLOBAL hM;        // manipulador de memória
  LPBYTE distImg;    // pontos da transformada distância

  // Inicialização da transformada-distância

  sImg = DimBitmap[nImgIn].cx*DimBitmap[nImgOut].cy;
  for (i=0; i<sImg; i++)
    PixImg[nImgOut][i] = PixImg[nImgIn][i];
  hM = GlobalAlloc(GHND, (DWORD)sImg);
  distImg = (LPBYTE) GlobalLock(hM);

  // Erode a imagem enquanto houver pixels > 0

  cont = 1;
  while (cont>0) {
    Erosao(nImgOut, nImgOut, np, strelem);
    for (i=0; i<sImg; i++)
      if (PixImg[nImgOut][i]==255)
        distImg[i]++;
    cont = 0;
    for (i=0; i<sImg; i++)
      cont+= PixImg[nImgOut][i];
  }

  // Identificação da transformada-distância com uma imagem em níveis de cinza

  maxD = 0;
  for (i=0; i<sImg; i++) {
    PixImg[nImgOut][i] = distImg[i];
    if (distImg[i] > maxD)
      maxD = distImg[i];
  }
  for (i=0; i<sImg; i++)
    PixImg[nImgOut][i]*= (BYTE)(255/maxD);

  GlobalUnlock(hM);

  return maxD;
}

/***************************************************************************************************************
 Esqueleto
 ---------
 Gera o esqueleto de uma imagem binária.
 Parâmetros:
   nImgIn : imagem binária original
   nImgOut: imagem esqueletonizada
 ***************************************************************************************************************/
void Esqueleto(int nImgIn, int nImgOut)
{
  int i;                // índice
  int sImg;             // tamanho da imagem
  BOOL not_OK;          // TRUE: quando a imagem estiver esqueletonizada
  int cont;             // contador de diferenças entre imagens
  HGLOBAL hM1,hM2;      // manipuladores de memória
  LPBYTE imgaux;        // imagem-buffer auxiliar
  LPBYTE imgAnt;        // imagem esqueletonizada do ciclo anterior
  #define nTolDif 10

  // Alocação de memória para imagens auxiliares e salvamento da imagem original

  sImg = DimBitmap[nImgIn].cx*DimBitmap[nImgIn].cy;
  hM1 = GlobalAlloc(GHND, (DWORD)sImg);
  imgaux = (LPBYTE) GlobalLock(hM1);
  hM2 = GlobalAlloc(GHND, (DWORD)sImg);
  imgAnt = (LPBYTE) GlobalLock(hM2);
  for (i=0; i<sImg; i++)
    imgaux[i] = PixImg[nImgIn][i];

  // Ciclo de esqueletonização

  not_OK = TRUE;
```

```
  int k = 1;
  while (not_OK) {
    // Efetua um ciclo de esqueletonização
    CicloEsquelet(nImgIn, nImgOut);
    // Apresentação da imagem esqueletonizada durante o ciclo corrente
    sprintf(NomeImg[nImgOut], "Ciclo de esqueletonização %d", k);
    SetBitmapBits(hbm[nImgOut], (DWORD)sImg, (LPSTR)PixImg[nImgOut]);
    MostraImagem(nImgOut);
    k++;
    // Compara imagem esqueletonizada atual com a do ciclo precedente
    cont = 0;
    for (i=0; i<sImg; i++)
      if (PixImg[nImgOut][i] != imgAnt[i])
        cont++;
    if (cont>nTolDif)   // excesso de diferenças: continua
      for (i=0; i<sImg; i++) {
        imgAnt[i] = PixImg[nImgOut][i];
        PixImg[nImgIn][i] = PixImg[nImgOut][i];
      }
      else              // ausência de diferenças: fim
        not_OK = FALSE;
  }

  // Recuperação da imagem original

  for (i=0; i<sImg; i++)
    PixImg[nImgIn][i] = imgaux[i];

  // Liberação de memória

  GlobalUnlock(hM1);
  GlobalUnlock(hM2);

  #undef nTolDif

  return;
}

/*****************************************************************************************************
 CicloEsquelet
 -------------
 Aplica um ciclo de esqueletonização a uma imagem binária utilizando vizinhança de Golay do tipo "L".
 Parâmetros:
   nImgIn : imagem original
   nImgOut: imagem após um ciclo de esqueletonização
 *****************************************************************************************************/
void CicloEsquelet(int nImgIn, int nImgOut)
{
  int i, j;          // índices
  int sImg;          // tamanho da imagem
  HGLOBAL hM;        // manipulador de memória
  LPBYTE imgaux;     // imagem-buffer auxiliar
  ponto hit [9];     // elemento estruturante para o "foreground"
  ponto miss[9];     // elemento estruturante para o "background"
  int nhit, nmiss;   // número de pontos dos elementos estruturantes hit e miss

  // Alocação de memória para imagem-auxiliar e salvamento da imagem original

  sImg = DimBitmap[nImgIn].cx*DimBitmap[nImgIn].cy;
  hM = GlobalAlloc(GHND, (DWORD)sImg);
  imgaux = (LPBYTE) GlobalLock(hM);
  for (i=0; i<sImg; i++)
    imgaux[i] = PixImg[nImgIn][i];

  //Ciclo de afinamento utilizando 8 instâncias (rotacionadas a cada 45 graus) das vizinhanças de Golay

  for (i=0; i<8; i++) {
    Load_Golay(i, hit, &nhit, miss, &nmiss);
    Thinning(nImgIn, nImgOut, nhit, hit, nmiss, miss);
    for (j=0; j<sImg; j++)
      PixImg[nImgIn][j] = PixImg[nImgOut][j];
  }

  // Recuperação da imagem original

  for (i=0; i<sImg; i++)
    PixImg[nImgIn][i] = imgaux[i];

  GlobalUnlock(hM);

  return;
}

/*****************************************************************************************************
```

```
 Load_Golay
 ----------
 Constrói uma vizinhança de Golay do tipo "L", rotacionada de um certo número indRot (0-7) de ângulos de
 45 graus.
 Parâmetros:
   indRot: índice da rotação de 45 graus
   hit   : elemento estruturante para o "foreground"
   nhit  : número de pontos de "hit"
   miss  : elemento estruturante para o "background"
   nmiss : número de pontos de "miss"
 ************************************************************************************************************/
void Load_Golay(int indRot, ponto *hit, int *nhit, ponto *miss, int *nmiss)
{
  int i,j,m,n;          // índices
  int pat[3][3];        // padrão utilizado para construir a vizinhança de Golay
  int nrot;             // número de rotações de 45 graus

  // Configuração inicial da vizinhança de Golay do tipo "L"

  if (indRot%2 == 0) {
    pat[0][0] = 1; pat[0][1] = 1; pat[0][2] = 1;
    pat[1][0] = 2; pat[1][1] = 1; pat[1][2] = 2;
    pat[2][0] = 0; pat[2][1] = 0; pat[2][2] = 0;
    nrot = indRot;
  }
  else {
    pat[0][0] = 2; pat[0][1] = 1; pat[0][2] = 2;
    pat[1][0] = 0; pat[1][1] = 1; pat[1][2] = 1;
    pat[2][0] = 0; pat[2][1] = 0; pat[2][2] = 2;
    nrot = indRot-1;
  }

  // Rotaciona o padrão original da vizinhança "E"

  for (i=1; i<=nrot; i++)
    rotPad(pat[0], pat[1], pat[2]);

  // Gera os elementos estruturantes do "foreground" e do "background"

  m = 0;
  n = 0;
  for (i=0; i<=2; i++)
    for (j=0; j<=2; j++)
      switch (pat[i][j]) {
        case 0:
          miss[m].x = i-1;
          miss[m].y = j-1;
          m++;
          break;
        case 1:
          hit[n].x = i-1;
          hit[n].y = j-1;
          n++;
          break;
        case 2:
          break;
      }
  *nhit = n;
  *nmiss = m;

  return;
}

/************************************************************************************************************
 rotPad
 ------
 Rotaciona de 45 graus um padrão representado por uma matriz quadrada 3x3.
 Parâmetros:
   a1,a2,a3: linhas da matriz
 ************************************************************************************************************/
void rotPad(int *a1, int *a2, int *a3)
{
  int i;
  int B[3][3];

  B[0][0] = a1[1];
  B[0][1] = a1[2];
  B[0][2] = a2[2];
  B[1][0] = a1[0];
  B[1][1] = a2[1];
  B[1][2] = a3[2];
  B[2][0] = a2[0];
  B[2][1] = a3[0];
  B[2][2] = a3[1];
```

```
  for (i=0; i<3; i++) {
    a1[i] = B[0][i];
    a2[i] = B[1][i];
    a3[i] = B[2][i];
  }

  return;
}

/*****************************************************************************************************
 Thinning
 --------
 Afinamento de imagem binária utilizando os elementos estruturantes strelem1 ("foreground") e strelem2
 ("background").
 Parâmetros:
   nImgIn  : imagem binária original
   nImgOut : imagem binária afinada
   strelem1: elemento estruturante para o "foreground"
   np1     : número de pontos de strelem1
   strelem2: elemento estruturante para o "background"
   np2     : número de ponstos de strelem2
 *****************************************************************************************************/
void Thinning(int nImgIn, int nImgOut, int np1, ponto *strelem1, int np2, ponto *strelem2)
{
  int i;        // índice
  int sImg;     // tamanho da imagem

  sImg = DimBitmap[nImgIn].cx*DimBitmap[nImgIn].cy;

  // Aplica Hit-Miss

  Hit_Miss(nImgIn, nImgOut, np1, strelem1, np2, strelem2);

  // Subtrai da imagem original pontos selecionados através da operação hit-miss

  for (i=0; i<sImg; i++)
    if ((PixImg[nImgIn][i]==255) && (PixImg[nImgOut][i]==0))
      PixImg[nImgOut][i] = 255;
    else
      PixImg[nImgOut][i] = 0;

  return;
}

/*****************************************************************************************************
 compact_buffer
 --------------
 A partir da imagem original limiarizada nImg8bits (contendo apenas os níveis de cinza 0 e 255), produz uma
 imagem bImg puramente binária.
 Parâmetros:
   nImg8bits: no. da imagem limiarizada representada em 8 bits
   bImg     : imagem binária correspondente
 *****************************************************************************************************/
void compact_buffer(int nImg8bits, BYTE *bImg)
{
  int size;     // tamanho da imagem original em 8 bits
  int bsize;    // tamanho da imagem compactada
  int i;        // índice

  // Inicialização da imagem compactada

  size  = DimBitmap[nImg8bits].cx*DimBitmap[nImg8bits].cy;
  bsize = size/8;
  for (i=0; i< bsize; i++)
    bImg[i] = 0;

  // Compactação da imagem

  for (i=0; i<size; i++)
    if (PixImg[nImg8bits][i]>0)
       switch (i%8) {
         case 0: bImg[i/8] |= 0200; break; // OR 10000000
         case 1: bImg[i/8] |= 0100; break; // OR 01000000
         case 2: bImg[i/8] |= 040 ; break; // OR 00100000
         case 3: bImg[i/8] |= 020 ; break; // OR 00010000
         case 4: bImg[i/8] |= 010 ; break; // OR 00001000
         case 5: bImg[i/8] |= 04  ; break; // OR 00000100
         case 6: bImg[i/8] |= 02  ; break; // OR 00000010
         case 7: bImg[i/8] |= 01  ; break; // OR 00000001
       }

  return;
}
```

```
/*****************************************************************************************************
 restore_buffer
 --------------
 A partir da imagem binária bImg produz uma imagem representada em 8 bits contendo apenas os níveis de
 cinza 0 (preto) e 255 (branco).
 Parâmetros:
   bImgIn    : imagem binária original
   nImg8bits: no. da imagem correspondente representada em 8 bits
 *****************************************************************************************************/
void restore_buffer(int nImg8bits, BYTE *bImg)
{
  int size;      // tamanho da imagem descompactada
  int i;         // índice

  // Inicialização da imagem em 8 níveis de cinza

  size = DimBitmap[nImg8bits].cx*DimBitmap[nImg8bits].cy;
  for (i=0; i<size; i++)
    PixImg[nImg8bits][i] = 0;

  // Descompactação da imagem binária

  for (i=0; i<size; i++)
    switch (i%8) {
      case 0: if (bImg[i/8] & 0200) PixImg[nImg8bits][i] = 255; break;
      case 1: if (bImg[i/8] & 0100) PixImg[nImg8bits][i] = 255; break;
      case 2: if (bImg[i/8] & 040)  PixImg[nImg8bits][i] = 255; break;
      case 3: if (bImg[i/8] & 020)  PixImg[nImg8bits][i] = 255; break;
      case 4: if (bImg[i/8] & 010)  PixImg[nImg8bits][i] = 255; break;
      case 5: if (bImg[i/8] & 04)   PixImg[nImg8bits][i] = 255; break;
      case 6: if (bImg[i/8] & 02)   PixImg[nImg8bits][i] = 255; break;
      case 7: if (bImg[i/8] & 01)   PixImg[nImg8bits][i] = 255; break;
    }

  return;
}

/*****************************************************************************************************
 img_union
 ---------
 Efetua a união lógica de duas imagens binárias.
 Parâmetros:
   bImg1, bImg2: imagens binárias dadas
   bImg3       : imagem resultante
 *****************************************************************************************************/
void img_union(BYTE *bImg1, BYTE *bImg2, int size, BYTE *bImg3)
{
  int i;

  for (i=0; i<size; i++)
     bImg3[i] = bImg1[i] | bImg2[i];

  return;
}

/*****************************************************************************************************
 img_and
 -------
 Efetua a intersecção de duas imagens binárias.
 Parâmetros:
   bImg1, bImg2: imagens binárias dadas
   bImg3       : imagem resultante
 *****************************************************************************************************/
void img_and(BYTE *bImg1, BYTE *bImg2, int size, BYTE *bImg3)
{
  int i;

  for (i=0; i<size; i++)
    bImg3[i] = bImg1[i] & bImg2[i];

  return;
}

/*****************************************************************************************************
 img_translate
 -------------
 Aplica translação horizontal, vertical ou combinada a uma imagem binária bImg1, segundo o vetor (x,y).
 Parâmetros:
   nImg1: número da imagem original não-binária
   bImg1: imagem binária original
   bImg2: imagem binária transladada
   sbImg: tamanho da imagem
   x,y  : coordenadas do vetor de translação
```

```
 Observação:
   x aponta para baixo e y para a direita da tela
 ************************************************************************************************************/
void img_translate(int nImg1, BYTE *bImg1, int x, int y, BYTE *bImg2)
{
  int i;                // índice
  int width, height;    // largura e altura da imagem original não binária
  int sbImg;            // tamanho da imagem original (e final) binária

  width  = DimBitmap[nImg1].cx;
  height = DimBitmap[nImg1].cy;
  sbImg  = width*height/8;

  if ((x==0) && (y==0))         // ausência de translação
    for (i=0; i<sbImg; i++)
      bImg2[i]= bImg1[i];
  else if ((x==0) && (y!=0))    // translação horizontal
    h_trans(bImg1, bImg2, y, width/8, height);
  else if ((y==0) && (x!=0))    // translação vertical
    v_trans(nImg1, bImg1, bImg2, x);
  else {                        // translação composta
    h_trans(bImg1, bImg2, y, width/8, height);
    bImg1 = bImg2;
    v_trans(nImg1, bImg1, bImg2, x);
  }

  return;
}

/************************************************************************************************************
 h_trans
 -------
 Aplica uma translação lateral "p" a uma imagem binária. Se p>0 o deslocamento é à direita; se p<0 é à
 esquerda.
 Parâmetros:
   bImg1: imagem binária original
   bImg2: imagem binária deslocada horizontalmente
   p    : deslocamento
   bw   : largura da imagem binária
   h    : altura da imagem não-binária correspondente
 ************************************************************************************************************/
void h_trans(BYTE *bImg1, BYTE *bImg2, int p, int bw, int h)
{
  int i,j;                  // índices
  HGLOBAL hM1, hM2;         // manipuladores de memória
  register BYTE *line;      // linha da imagem binária original
  register BYTE *sh_line;   // linha da imagem binária deslocada

  // Alocação de memória para as linhas da imagem

  hM1 = GlobalAlloc(GHND, (DWORD)bw);
  line = (BYTE *) GlobalLock(hM1);
  hM2 = GlobalAlloc(GHND, (DWORD)bw);
  sh_line = (BYTE *) GlobalLock(hM2);

  // Deslocamento horizontal das linhas da imagem

  for (i=0; i<h; i++) {
    for (j=0; j<bw; j++)
      line[j] = bImg1[i*bw+j];
    if (p>0)
      rshiftstr(line, sh_line, p, bw);
    else
      lshiftstr(line, sh_line, -p, bw);
    for (j=0; j<bw; j++)
      bImg2[i*bw+j] = sh_line[j];
  }

  // Liberação de memória

  GlobalUnlock(line);
  GlobalUnlock(sh_line);

  return;
}

/************************************************************************************************************
 rshiftstr
 ---------
 Aplica a um string de caracteres de comprimento r um deslocamento p à direita.
 Parâmetros:
   str1: string original
   str2: string deslocado
   r   : tamanho do string
```

```
   p    : deslocamento
 ***************************************************************************************************************/
void rshiftstr(BYTE *str1, BYTE *str2, int p, int r)
{
  int i;              // índice
  int m;              // índice do byte
  int n;              // índice do bit
  byte left;          // bits de ordem superior
  byte old_right;     // bits de ordem inferior do último byte

  for (i=0; i<r; i++)
    str2[i] = 0;

  m = p/8;
  n = p%8;

  old_right = 0;

  for (i=0; i<r-m; i++) {
    left = str1[i] >> n;
    if (i>0)
      old_right = str1[i-1] << 8-n;
    str2[i+m] = old_right | left;
  }

  return;
}

/***************************************************************************************************************
 lshiftstr
 ---------
 Aplica a um string de caracteres de comprimento r um deslocamento p à esquerda.
 Parâmetros:
   str1: string original
   str2: string deslocado
   r   : tamanho do string
   p   : deslocamento
 ***************************************************************************************************************/
void lshiftstr(BYTE *str1, BYTE *str2, int p, int r)
{
  int i;              // índice
  int m;              // índice do byte
  int n;              // índice do bit
  byte right;         // bits de ordem inferior
  byte old_left;      // bits de ordem superior do último byte

  for (i=0; i< r; i++)
    str2[i] = 0;

  m = p/8;
  n = p%8;

  old_left = 0;

  for (i=r-1; i>=m; i--) {
    right = str1[i] << n;
    if (i<r-1)
      old_left = str1[i+1] >> 8-n;
    str2[i-m] = old_left | right;
  }

  return;
}

/***************************************************************************************************************
 v_trans
 -------
 Aplica uma translação vertical "x" a uma imagem binária. Esta operação é equivalente a uma translação de
 "-x" da imagem binária transposta seguida de uma transposição da imagem resultante, de modo a representá-la
 da forma normal, isto é, "por linhas".
 Parâmetros:
   nImg1: número da imagem não-binária original
   bImg1: imagem binária original
   bImg2: imagem binária transladada na vertical
   x    : translação vertical
 ***************************************************************************************************************/
void v_trans(int nImg1, BYTE *bImg1, BYTE *bImg2, int x)
{
  int width, height;    // largura e altura da imagem não-binária original
  int bsImg;            // tamanho da imagem binária
  HGLOBAL hM1, hM2;     // manipuladores de memória
  BYTE *bImg3, *bImg4;  // imagens binárias auxiliares

  // Alocação de memória
```

```
  width  = DimBitmap[nImg1].cx;
  height = DimBitmap[nImg1].cy;
  bsImg = width*height/8;
  hM1 = GlobalAlloc(GHND, (DWORD)bsImg);
  bImg3 = (BYTE *) GlobalLock(hM1);
  hM2 = GlobalAlloc(GHND, (DWORD)bsImg);
  bImg4 = (BYTE *) GlobalLock(hM2);

  // Translação da imagem transposta e retorno à forma original

  transpose(bImg1, bImg3, height, width);
  h_trans(bImg3, bImg4, x, height/8, width);
  transpose(bImg4, bImg2, width, height);

  // Liberação de memória

  GlobalUnlock(hM1);
  GlobalUnlock(hM2);

  return;
}

/*****************************************************************************************************
 transpose
 ---------
 Transpõe a imagem binária, substituindo linhas por colunas e vice-versa.
 Exemplo:    À esquerda, imagem original; à direita imagem transposta:
    00011100 ...                01000101 ...
    10101010 ...                00001101 ...
    00010010 ...                01000101 ...
    00000000 ...                10100101 ...
    01001010 ...                11001001 ...
    11110111 ...                10000110 ...
    00100100 ...                01101101 ...
    11111011 ...                00000101 ...
             ...                         ...
 Parâmetros:
   bImg1: imagem binária original
   bImg2: imagem binária transposta
   h    : número de linhas da imagem não-binária original
   w    : número de colunas da imagem não-binária original
 *****************************************************************************************************/
void transpose(BYTE *bImg1, BYTE *bImg2, int h, int w)
{
  int i,j;       // indices
  int last;      // índice offset
  int bw, bh;    // largura e altura da imagem binária
  int m;         // índice do bit
  int n;         // índice do byte
  HGLOBAL hM;    // manipulador de memória
  BYTE *column; // coluna de bytes de uma imagem descrita "por linhas"

  bw = w/8;
  bh = h/8;
  hM = GlobalAlloc(GHND, (DWORD)h);
  column = (BYTE *) GlobalLock(hM);

  last = 0;
  for (i=0; i<bw; i++) {
    for (j=0; j<h; j++)
      column[j] = bImg1[i+bw*j];
    n = 0;
    for (j=0; j<h; j++) {
      m = j/bh;
      bImg2[last+j] = v_byte(&column[n*8], m);
      n++;
      if (n==bh)
        n = 0;
    }
    last = last+h;
  }

  GlobalUnlock(hM);

  return;
}

/*****************************************************************************************************
 v_byte
 ------
 Dado um string de bytes, esta função produz um novo byte selecionando, de cada byte do string original, o
 m-ésimo bit. Por exemplo, considerando-se o seguinte string de bytes:
 00011111 10101010 00111100 00000000 11100011 11111111 00001000 11110001 ...
```

```
  *       *       *       *       *       *       *       *
e m = 2, o byte resultante será: 01101101
Parâmetros:
  str1: string de bytes
  m   : ordem dos bits a serem selecionados
Retorna:
  byte com os bits de ordem m de cada byte do string str1
*****************************************************************************************************/
BYTE v_byte(BYTE *str1, int m)
{
  int  i;    // índice
  BYTE ch;   // byte construído
  BYTE mask; // máscara binária usada para selecionar o bit de ordem m de cada byte

  switch (m) {
    case 0: mask = 0200; break; // 10000000
    case 1: mask = 0100; break; // 01000000
    case 2: mask = 040 ; break; // 00100000
    case 3: mask = 020 ; break; // 00010000
    case 4: mask = 010 ; break; // 00001000
    case 5: mask = 04  ; break; // 00000100
    case 6: mask = 02  ; break; // 00000010
    case 7: mask = 01  ; break; // 00000001
  }

  ch = 0;

  for (i=0; i<8; i++) {
    if (m>i)
      ch |= (str1[i] & mask) << m i;
    else
      ch |= (str1[i] & mask) >> i-m;
  }

  return(ch);
}

/*****************************************************************************************************
 ALG-VISAOCOMP.H
 ---------------

 Definições e declarações de tipos e variáveis utilizadas pelo módulo ALG-VISAOCOMP.CPP
 *****************************************************************************************************/

#define STRICT
#include <windows.h>
#include <windowsx.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

/*****************************************************************************************************
  DEFINIÇÕES
 *****************************************************************************************************/

#define MAXSTRINGSIZE 150                          // tamanho máximo de string
#define NMAX_IMG 20                                // número máximo de imagens alocadas
#define N_PixelsX 640                              // número máximo de colunas da imagem
#define N_PixelsY 480                              // número máximo de linhas da imagem
#define NMAX_RETAS_IMG 100                         // número máximo de retas da imagem

/*****************************************************************************************************
 VARIÁVEIS UTILIZADAS PELOS OBJETOS GRÁFICOS DE INTERFACE
 *****************************************************************************************************/

extern HWND MainWindow;                            // manipulador da janela da aplicação
extern HBITMAP hbm[NMAX_IMG];                      // manipuladores de imagem armazenada
extern HBITMAP hImgTmp;                            // manipulador de imagem temporária
BYTE ConvIndPal[256] =                             // tabela de conversão de índices de cor da palette
    {  0,  0,  0,  0,  0,  0, 11, 11, 11, 11, 11,
      11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
      21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
      31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
      41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
      51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
      61, 62, 63, 64, 65, 66, 67, 68, 69, 70,
      71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
      81, 82, 83, 84, 85, 86, 87, 88, 89, 90,
      91, 92, 93, 94, 95, 96, 97, 98, 99,100,
     101,102,103,104,105,106,107,108,109,110,
     111,112,113,114,115,116,117,118,119,120,
     121,122,123,124,125,126,127,128,129,130,
     131,132,133,134,135,136,137,138,139,140,
```

```
    141,142,143,144,145,146,147,148,149,150,
    151,152,153,154,155,156,157,158,159,160,
    161,162,163,164,165,166,167,168,169,170,
    171,172,173,174,175,176,177,178,179,180,
    181,182,183,184,185,186,187,188,189,190,
    191,192,193,194,195,196,197,198,199,200,
    201,202,203,204,205,206,207,208,209,210,
    211,212,213,214,215,216,217,218,219,220,
    221,222,223,224,225,226,227,228,229,230,
    231,232,233,234,235,236,237,238,239,240,
    241,242,243,244,245,245,245,245,245,245,
    255,255,255,255,255 };

/*****************************************************************************************************
 VARIÁVEIS ASSOCIADAS A IMAGENS
 *****************************************************************************************************/

extern BYTE PixImg[NMAX_IMG][N_PixelsX*N_PixelsY];  // imagens
extern SIZE DimBitmap[NMAX_IMG];                    // dimensões das imagens
extern char NomeImg[NMAX_IMG][MAXSTRINGSIZE];       // nomes das imagens
extern int IndImgSel;                               // número da imagem selecionada
extern int UltImg;                                  // última imagem
extern double SigmaN;                               // desvio-padrão do ruído Gaussiano da imagem
struct {                                            // Coordenadas das Bordas de Imagem Segmentada
  float XBorda[4*N_PixelsX*N_PixelsY];              // abcissas das bordas
  float YBorda[4*N_PixelsX*N_PixelsY];              // ordenadas das bordas
} CoordBordas;                                      // coordenadas das bordas da última imagem
extern BYTE imgRec[4*N_PixelsX*N_PixelsY];          // última imagem reconstruída

/*****************************************************************************************************
 VARIÁVEIS ASSOCIADAS A FEIXES DE RETAS
 *****************************************************************************************************/

extern struct {
  int nRetasImg;                                    // número de retas identificadas na imagem
  BOOL tipo;                                        // TRUE: feixe linear ; FALSE: feixe angular
  double Raio  [NMAX_RETAS_IMG];                    // distância da reta medida a partir da origem(0,0)
  double Angulo[NMAX_RETAS_IMG];                    // orientação da reta relativamente ao eixo X
  double Medida[NMAX_RETAS_IMG];                    // medida linear ou angular da escala-objeto associada
  double MedCal[NMAX_RETAS_IMG];                    // medida linear ou angular calculada
  double Sigma [NMAX_RETAS_IMG];                    // desvio-padrão da reta média ajustada
  int    nPts  [NMAX_RETAS_IMG];                    // número de pontos da reta
} FeixeRetas[NMAX_IMG];                             // feixe de retas

/*****************************************************************************************************
 VARIÁVEIS PARA CALIBRAÇÃO DE MICROESCALAS
 *****************************************************************************************************/

extern double DistTracos;                           // distância entre os traços da escala de calibração
extern int NumTracos;                               // número de tracos calibrados na escala
extern struct{
  double nominal[NMAX_RETAS_IMG];                   // medidas nominais
  double real [NMAX_RETAS_IMG];                     // medidas reais
} Escala;                                           // medidas de uma microescala linear

/*****************************************************************************************************
 TIPOS E VARIÁVEIS DE USO GERAL
 *****************************************************************************************************/

typedef struct {int x; int y;} ponto;               // coordenadas (x,y) de um ponto
double PI = acos(-1.0);                             // PI
double GRAD = PI/(double)180.0;                     // conversão graus->radianos

/*****************************************************************************************************
 FUNÇÕES
 *****************************************************************************************************/

void ApagaImagem(int nImg);
void Ordena(int n, double *X, double *Y, double *Xord, double *Yagr);
void DetCoordBordas(int nImgIn);
void Op_BOOL(int numOp, int nImg1, int nImg2, int nImgOut);
void LimiarFixo(int nImgIn, int nImgOut, int limiar, BOOL Bin);
void FiltroMedia(int nImgIn, int nImgOut, int largJan);
void FiltroMediana(int nImgIn, int nImgOut, int largJan);
void ImagemdeRetas(int nImgIn, int nImgOut, int nRetas, double *Theta, double *Rho);
void FeixeMedio(int nImg1, int nImg2, int nImgOut);
void AssociaImagemObjeto(int nImg, BOOL fLinear, BOOL medReal);
void Calibracao(int nImg1, int nImg2);
void AfinaBordas(LPBYTE imgRec, int nImg, float *G, unsigned char *D, BOOL Rec);
void ApresentaImgBordas(int nCol2, int nLin2, LPBYTE imgBordas, int nImgIn, int nImgOut, BOOL Rec, int tB);
void CalculaMedidas(int nImg, int tObjeto);
void Reconstrucao(int nImgIn, int nImgOut);
void DBordasRoberts(int nImgIn, int nImgOut, int tipBorda, BOOL Bin, double pcFBordas,BOOL Rec);
void DBordasSobel(int nImgIn, int nImgOut, int tipBorda, BOOL Bin, double pcFBordas, BOOL Rec);
```

```
void DBordasSubPixel5x5(int nImgIn, int nImgOut, int tipBorda, BOOL Bin, double pcFBordas, BOOL Rec);
void DBordasSubPixel3x3(int nImgIn, int nImgOut, int tipBorda, BOOL Bin, double pcFBordas, BOOL Rec);
void TransHough(int nImgIn, int nImgOut, BOOL Dup, BOOL tipoImg, BOOL aut, double aMin, double aMax,
     double dAng, double L, double E, BOOL rPar, int dPic, BOOL nfxP, int nPic, double Fcorte);
void CalcParamTHough(int nImgIn, double ThetaMin, double ThetaMax, double dTheta, double L, double b,
     int *nTheta, int *nRho, double *RhoMin, double *RhoMax, double *dRho);
void AcumulaHough(int nImgIn, BOOL imgBin, double *H, int nTheta, double ThetaMin, double dTheta,
     int nRho, double RhoMin, double RhoMax);
void AchaRetasParalelas(int nImg,double *H,int nTheta,double ThetaMin,double dTheta,int nRho,double RhoMin,
     double dRho,double b,BOOL nfxPic,int nPic,double Fcorte);
void AchaRetas(int nImg,double *H,int nTheta,double ThetaMin,double dTheta,int nRho,double RhoMin,
     double dRho,double b,BOOL nfxPic,int nPic,double Fcorte);
void AjustaRetas1(int nImgBor, int nImgRtAp, int nImgRtAj, double largJan, BOOL Dup);
void AjustaRetas2(int nImgBor, int nImgRtAp, int nImgRtAj, double largJan, BOOL Dup);
void Dilatacao(int nImgIn, int nImgOut, int np, ponto *strelem);
void CriaElemEst5Pts(double r1, double ang1, double r2, double ang2, ponto *strelem);
void CriaElemEst3Pts(double r, double ang, ponto *strelem);
void Erosao(int nImgIn, int nImgOut, int np, ponto *strelem);
void Hit_Miss(int nImgIn, int nImgOut, int np1, ponto *strelem1, int np2, ponto *strelem2);
void Abertura(int nImgIn, int nImgOut, int np, ponto *strelem);
void Fechamento(int nImgIn, int nImgOut, int np, ponto *strelem);
void DilatacaoCond(int nImgIn, int nImgRef, int nImgOut, int np, ponto *strelem);
void DestroiObjFront(int nImgIn, int nImgOut, int np, ponto *strelem, int m1, int m2);
void Esqueleto(int nImgIn, int nImgOut);
void CicloEsquelet(int nImgIn, int nImgOut);
void Load_Golay(int index, ponto *hit, int *nhit, ponto *miss, int *nmiss);
void rotPad(int *a1, int *a2, int *a3);
void Thinning(int nImgIn, int nImgOut, int np1, ponto *strelem1, int np2, ponto *strelem2);
void compact_buffer(int nImg8bits, BYTE *bImg);
void restore_buffer(int nImg8bits, BYTE *bImg);
void img_translate(int nImg, BYTE* bImg, int x, int y, BYTE *t_bImg);
void img_union(BYTE *bImg1, BYTE *bImg2, int sbImg, BYTE *bImg3);
void img_and(BYTE *bImg1, BYTE *bImg2, int sbImg, BYTE *bImg3);
void h_trans(BYTE *bImg1, BYTE *bImg2, int y, int bwidth, int height);
void v_trans(int nImg1, BYTE *bImg1, BYTE *bImg2, int x);
void rshiftstr(BYTE *str1, BYTE *str2, int p, int r);
void lshiftstr(BYTE *str1, BYTE *str2, int p, int r);
void transpose(BYTE *bImg1, BYTE *bImg2, int h, int w);
void jacobi(double **a, int n, double *d, double **v, int *nrot);
void eigsrt(double *Lambda, double **E, int m);
void ludcmp(double **a, int n, int *indx, double *d);
void lubksb(double **a, int n,int *indx, double *b);
void mprove(double **a, double **alud, int n, int *indx, double *b, double *x);
void balanc(double **a, int n);
void elmhes(double **a, int n);
void hqr(double **a, int n, double *wr, double *wi);
void sort(int n, double *ra);
void spline(float *x, float *y, int n, float yp1, float ypn, float *y2);
void splint(float *xa, float *ya, float *y2a, int n, float x, float *y);
void free_dvector(double *v, int nl, int nh);
BYTE v_byte(BYTE *str1, int m);
BOOL ImagemBinaria(int nImg);
BOOL InterRetas(int nImg1, double Ang1, double R1, double Ang2, double R2, double *x,double *y);
unsigned char CalcOriGrad(double Gx, double Gy, double ang);
int CalcLimiar(int nImgIn);
int Rotulacao(int nImgIn, int nImgOut, double minArea);
int MedOrdpRot(int nImg1, int nImg2, int nImg3);
int InterFeixes(int nImg1, int nImg2, double* X, double *Y);
int ClassificaRetas(double Angulo, double Raio, double *a, double *b);
int CalcCompReta(int nImg, int nreta);
int CalcLimiarHadd(double pFBordas, double fator);
int CalcParImgBordas(int nImgIn, BOOL Rec, int *nCol, int *nLin, float *coef);
int SelecionaPontos(int nImgBor, BOOL Dup, double largJan, double raio, double angulo, float *X, float *Y);
int SelecionaPontos2(int nImg1, int nImg2, int nmask, float *X, float *Y);
int CriaElemEstCirc(int Raio, ponto *ElemEst);
int CriaElemEstAro(int Raio, ponto *ElemEst);
int CriaElemEstRet(float larg, float alt, ponto *ElemEst);
int CriaElemEstAroRet(float larg, float alt, ponto *ElemEst);
int TransfDist(int nImgIn, int nImgOut, int np, ponto *strelem);
int *ivector(int nl, int nh);
float CalcMagGrad1(double Gx, double Gy);
float CalcMagGrad2(double Gx, double Gy);
float CalcMagGrad3(double Gx, double Gy);
float *vector(int nrl, int nrh);
float **matrix(int nrl, int nrh, int ncl, int nch);
double EstatRuido(int nImg);
double CalcCentroide(double *X, int n, int iMax);
double AchaMaxLoc1D(double *y, int n, int b, double *x);
double AchaMaxLoc2D(double *z, int nx, int ny, int b, double *x, double *y);
double MaxVetor(int n, double *X);
double IdFreqESPRIT(int nImgIn, int nFreq, BOOL Vert);
double *dvector(int nl, int nh);
double **dmatrix(int nrl, int nrh, int ncl, int nch);
```

```
/*****************************************************************************************************
 FUNÇÕES EXTERNAS
 *****************************************************************************************************/

extern void MostraImagem(int nImg);
extern void ImagemdeRetas(int nImgIn, int nImgOut, int nRetas, double *Theta, double *Rho);
extern void InfoCaractImg(int nImgOut, int caso, int nc);
extern void MostraWinDisp(int nImg, LONG width, LONG height, char *title);
```