# Um método escalável de reconhecimento de objetos com uso de informações estruturais de grafos-chave

Estephan Dazzi Wandekoken

Tese apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Doutor em Ciências

Doutorado em Ciência da Computação
Orientador: Prof. Dr. Roberto Marcondes Cesar - Jr.

Este trabalho foi desenvolvido com suporte do CNPq, da CAPES e da FAPESP.

São Paulo, junho de 2015

# Um método escalável de reconhecimento de objetos com uso de informações estruturais de grafos-chave

Esta é a versão original da tese elaborada pelo
candidato (Estephan Dazzi Wandekoken), tal como
submetida à Comissão Julgadora.

# Resumo

DAZZI, E. **Um método escalável de reconhecimento de objetos com uso de informações estruturais de grafos-chave**. 2015. 85 f. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2015.

Uma abordagem de sucesso para o problema de reconhecimento de objetos envolve a extração, a partir de imagens de teste e de treinamento, de características locais invariantes a transformações visuais, denominadas pontos-chave. Correspondências são então criadas entre as imagens mediante o estabelecimento de correspondências entre essas características locais. Atualmente, a maioria dos métodos na literatura emprega a estratégia de estabelecer correspondências entre pontos-chave individuais, isto é, uma correspondência individual entre um ponto-chave na imagem de teste e um ponto-chave em uma imagem de treinamento. Neste trabalho, é apresentado um método de reconhecimento de objetos que emprega correspondências entre pequenos grafos locais de pontos-chave, denominados grafos-chave, em vez de correspondências simples entre pontos-chave individuais. Um grafo-chave consiste em um circuito orientado (um grafo) com três vértices, em que cada vértice é um ponto-chave. Para que uma correspondência entre grafos-chave seja considerada válida, os descritores dos vértices (pontos-chave) devem ser similares, assim como ambos os grafos-chave devem satisfazer propriedades estruturais relacionadas à orientação, escala, posicionamento relativo e sentido de ciclo dos pontos-chave; exigir a satisfação dessas propriedades estruturais permite a correta exclusão da grande maioria das correspondências entre pontos-chave inicialmente estabelecidas. Uma das duas principais contribuições deste trabalho consiste nesse conjunto heterogêneo de informações estruturais usadas para filtrar correspondências entre pontos-chave. A outra contribuição principal deste trabalho é um algoritmo eficiente para realizar a seleção de triplas de pontos-chave na imagem de teste; cada uma das triplas produzidas é utilizada como um grafo-chave. O algoritmo baseia-se no uso de triangulações de Delaunay complementares, e produz uma quantidade de triplas que aumenta linearmente com o número de pontos-chave na imagem de teste. Em seguida, são estabelecidas correspondências entre grafos-chave na imagem de teste e pontos-chave nas imagens de treinamento (os quais estão armazenados em uma árvore indexadora); isso produz correspondências entre grafos-chave. Por fim, cada correspondência entre grafos-chave é utilizada para avaliar uma pose de objeto candidata (uma transformação afim entre as imagens). O método proposto neste trabalho foi avaliado em uma tarefa de reconhecimento de objetos e estimação de pose. Dois cenários foram investigados: um cenário em que relativamente poucas imagens de treinamento (250) foram empregadas e outro cenário que utiliza uma quantidade muito maior de imagens de treinamento. Os resultados experimentais mostram que o método proposto neste trabalho alcançou um desempenho superior em ambos os cenários, em comparação a outros métodos no estado-da-arte.

**Palavras-chave:** Características locais, grafos, reconhecimento de objetos.

# Abstract

DAZZI, E. **Scalable object recognition using keygraph structural information**. 2015. 120 f. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2015.

A succesful approach to object recognition involves extracting, from query and training images, invariant local image features (keypoints); image matching then proceeds by matching those local image features. Currently, most methods in the literature follow the strategy of establishing correspondences between one-to-one keypoints. This work presents an object recognition method that employs matches between local graphs of keypoints, called keygraphs, instead of simple keypoint matches. A keygraph consists in an oriented circuit (a graph) with three vertices, where each vertex is a keypoint. For a keygraph match to be valid, the vertices (keypoints) descriptors must be similar and both keygraphs must satisfy structural properties concerning keypoints orientation, scale, relative position and cyclic ordering; this allows to correctly filter out the large majority of the initial keypoint matches. This heterogeneous set of structural information for keypoint match filtering is the first main contribution of this work. The second main contribution is an algorithm to smartly sample triples of keypoints (*i.e.*, keygraphs) in a query image, which is based on the use of complementary Delaunay triangulations. The algorithm generates a number of triples that grows linearly with the number of keypoints in the query image. Query keygraphs are then matched against the indexed training keypoints; each established keygraph match is used to evaluate a candidate pose (an affine transformation). The proposed method has been evaluated for object recognition and pose estimation. Two scenarios were evaluated: using a relatively small number of training images (250) or using a much larger set of training images. The proposed method achieved a better performance in comparison to state-of-the art methods, in both scenarios.

**Keywords:** Local features, Graphs, object recognition.

# Sumário

# Lista de Figuras

# Capítulo 1

# Introdução

## 1.1 Considerações preliminares

Um problema clássico em visão computacional consiste em decidir se um padrão visual específico está presente em uma imagem. Se esse padrão visual que é buscado for um objeto, o problema considerado será, então, o *reconhecimento de objetos*.

Uma abordagem de sucesso para realizar o reconhecimento de objetos envolve encontrar correspondências existentes entre características visuais locais de imagens. Correspondências são estabelecidas entre uma imagem do objeto que se deseja encontrar (imagem de treinamento) e uma cena qualquer na qual esse objeto pode estar presente (imagem de teste). Além de detectar a ocorrência de um objeto, é desejável que o método de reconhecimento de objetos também seja capaz de estimar, com alta precisão, a pose do objeto encontrado; nesse contexto, a pose de um objeto consiste numa combinação da sua posição e da sua orientação na imagem de teste.

Para que seja possível detectar muitos tipos de objetos diferentes, sob diferentes ângulos de visão, o problema de reconhecimento de objetos pode ser tratado mediante o estabelecimento de correspondências entre uma imagem de teste e um *conjunto* de imagens de treinamento, em que cada imagem de treinamento apresenta um ponto-de-vista de um objeto a ser encontrado.

Na atualidade, uma abordagem de sucesso para o problema de estabelecer correspondências entre imagens baseia-se na extração, em cada imagem, de características visuais locais denominadas *pontos-chave*. Cada ponto-chave é localizado em uma região com uma grande quantidade de informação visual; então, os *pixels* nessa região são usados para calcular um vetor de características, denominado *descritor*, que representa a informação visual na região da imagem em que o ponto-chave foi localizado. Dessa forma, com cada imagem representada por um conjunto de pontos-chave, uma correspondência entre duas *imagens* ocorre quando há um número suficiente de correspondências entre os *pontos-chave* dessas duas imagens.

A detecção de pontos-chave e a extração de seus descritores são feitas de forma a proporcionar algum nível de invariância em resposta a transformações visuais comuns tais como mudanças de iluminação, de escala e de ângulo de visão. Isso possibilita que duas imagens de um mesmo objeto, sujeitas a diferentes transformações visuais, possam produzir correspondências entre seus pontos-chave.

Atualmente, a maioria dos métodos na literatura emprega a estratégia de estabelecer correspondências entre pontos-chave individuais, em que cada correspondência ocorre entre um ponto-chave individual na imagem de teste e um ponto-chave individual em uma imagem de treinamento. Entre-

tanto, o uso de correspondências entre pontos-chave individuais possui uma desvantagem inerente: limitar os mecanismos possíveis de exploração das informações estruturais existentes em vizinhanças de pontos-chave (uma vizinhança de pontos-chave é composta de pontos-chave próximos entre si em uma imagem).

Este trabalho apresenta uma abordagem aprimorada para lidar com a organização estrutural de pontos-chave em imagens. A ideia é substituir correspondências entre pontos-chave individuais por correspondências entre *pequenos grafos locais de pontos-chave*, um conceito que foi inicialmente proposto por Hashimoto e Cesar Jr (2009). Cada imagem é representada por um grande conjunto desses grafos locais de pontos-chave. Cada um desses grafos, denominados *grafos-chave*, possui vértices que consistem em pontos-chave obtidos em uma mesma imagem, e possui arestas que armazenam informações estruturais acerca desses vértices.

A vantagem de utilizar grafos-chave em vez de pontos-chave é que o estabelecimento de uma correspondência entre grafos-chave envolve não somente que os descritores dos vértices (pontos-chave) sejam semelhantes, mas também demanda que ambos os grafos (na imagem de teste e de treinamento) satisfaçam relações estruturais calculadas com uso de informações nas arestas e nos vértices dos grafos-chave. Assim, enquanto o estabelecimento de correspondências entre pontos-chave individuais utiliza somente a semelhança entre descritores e (possivelmente) alguma propriedade estrutural simples, o estabelecimento de correspondências entre grafos-chave, além de empregar semelhança entre descritores, também explora a *estrutura* de vizinhanças de pontos-chave.

Um grafo-chave consiste em um circuito orientado (um grafo) com três vértices, em que cada vértice é um ponto-chave. Para que uma correspondência entre grafos-chave seja considerada válida, os descritores dos vértices (pontos-chave) devem ser similares, assim como ambos os grafos-chave devem satisfazer propriedades estruturais relacionadas à orientação, escala, posicionamento relativo e sentido de ciclo dos pontos-chave. Exigir a satisfação dessas propriedades estruturais permite a correta exclusão da grande maioria das correspondências entre pontos-chave inicialmente estabelecidas. Isso possibilita que o método de reconhecimento de objetos seja capaz de alcançar um melhor desempenho, com um menor custo computacional.

## 1.2 Objetivos

O objetivo deste trabalho é descrever um novo método desenvolvido para o reconhecimento de objetos, em que grafos de pontos-chave são empregados, em vez de pontos-chave individuais. Modelos e algoritmos são desenvolvidos, e resultados experimentais extensivos mostram um desempenho comparável ao de métodos do estado-da-arte, com algumas vantagens específicas, como será mostrado nesta tese.

## 1.3 Contribuições

Este trabalho apresenta duas contribuições principais:

- Um algoritmo eficiente para selecionar um conjunto diverso de grafos-chave em uma imagem de teste. Cada grafo-chave consiste em uma tripla de pontos-chave. A seleção de triplas ocorre em duas etapas. Inicialmente, um subconjunto dos pontos-chave da imagem de teste é selecionado, de forma que pontos-chave selecionados não estejam demasiadamente próximos entre si na

imagem de teste. Na segunda etapa, uma triangulação de Delaunay é empregada, produzindo, assim, triplas de pontos-chave na imagem de teste. As triplas de pontos-chave geradas possuem propriedades favoráveis ao estabelecimento de correspondências: os pontos-chave chave que compõem uma tripla estão próximos entre si na imagem de teste, sem que as arestas dos grafos-chave gerados sejam demasiado curtas. O processo descrito para gerar as triplas – seleção de subconjunto de pontos-chave e uso de triangulação de Delaunay – é repetido $T$ vezes; uma estratégia é empregada para aumentar a diversidade das triplas produzidas. A complexidade do algoritmo de seleção de triplas é $O(Tn \log n)$, para uma imagem de teste com $n$ pontos-chave, e produz um total de $O(Tn)$ triplas de pontos-chave. Após a geração das triplas na imagem de teste, correspondências entre grafos-chave são estabelecidas, e cada uma dessas correspondência entre grafos-chave é usada para avaliar uma pose de objeto candidata (uma transformação afim).

- Uma estratégia para utilizar diferentes tipos de informações estruturais, que são usadas para avaliar se uma correspondência entre grafos-chave é valida. Inicialmente, correspondências candidatas entre grafos-chave são estabelecidas somente com base na similaridade entre os descritores dos pontos-chave que compõem as duas triplas (na imagem de teste e na de treinamento). Em seguida, ambos os grafos-chave devem satisfazer as propriedades estruturais para que a correspondência entre eles seja considerada válida. Para que essas propriedades sejam satisfeitas, é necessário que os pontos-chave envolvidos na correspondência entre as triplas variem consistentemente entre a imagem de teste e a de treinamento, em relação ao posicionamento relativo, ao sentido de ciclo dos circuitos e aos valores de escala e de orientação dos pontos-chave (vértices dos grafos). Como resultado, mais de 99% das correspondências entre grafos-chave inicialmente estabelecidas são corretamente eliminadas, o que melhora tanto o desempenho (pois correspondências falsas são excluídas) quanto a eficiência (pois será necessário avaliar menos poses candidatas no estágio final de estimação de pose).

## 1.4 Organização deste trabalho

As seções seguintes deste trabalho estão organizadas como segue. O Capítulo 2 apresenta uma revisão dos principais conceitos básicos envolvidos nesta tese, assim como uma discussão acerca da literatura relevante. O Capítulo 3 descreve o método proposto para o reconhecimento de objetos. O Capítulo 4 apresenta os resultados experimentais obtidos a partir da avaliação sistemática do método proposto e da comparação com outros métodos no estado-da-arte. Esta tese é concluída no Capítulo 5, onde são feitos comentários sobre futuros desenvolvimentos.

# Capítulo 2

# Conceitos

## 2.1 Características locais para o reconhecimento de objetos

Estabelecer casamentos entre imagens é um problema fundamental em visão computacional. Diferentes tarefas em visão computacional demandam o emprego de um método para estimar a similaridade entre imagens. Por exemplo, o reconhecimento de objetos ou de ambientes, a determinação da estrutura tridimensional de um objeto a partir de diferentes pontos-de-vista (*structure-from-motion*), a determinação da geometria epipolar entre um par de imagens diferentes de uma mesma cena (estéreo) e o rastreamento de objetos que se movem (rastreamento através uma sequência de imagens).

Uma distinção conceitual fundamental existente na tarefa de reconhecer objetos em imagens envolve o tipo de representação visual que é utilizada: uma descrição global de cada imagem ou um conjunto de representações locais para cada imagem. Uma vantagem inerente que características locais possuem manifesta-se nos casos em que os objetos desejados estão somente parcialmente visíveis na imagem de teste, pois correspondências são estabelecidas apenas entre (pequenas) regiões das imagens.

O uso de características locais para realizar o reconhecimento de objetos se distingue da abordagem que emprega uma representação global das informações visuais em cada imagem. Como um exemplo de representação global de imagens, é possível considerar uma imagem completa como um elemento único a ser classificado, representado como um vetor de características; a intensidade de cada *pixel* da imagem tornaria-se um atributo desse vetor de características. Então, métodos classificadores de padrões (como redes neurais) seriam empregados para encontrar os objetos, mediante a classificação de uma imagem como sendo da classe "com objeto" ou da classe "sem objeto". Entretanto, essa abordagem global apresenta problemas nos casos em que o objeto buscado está somente parcialmente visível na imagem de teste. Em comparação, o método das características visuais locais lida naturalmente com esse problema de visibilidade parcial, já que correspondências são estabelecidas somente entre áreas pequenas (locais) das imagens.

A Figura 3.1 apresenta as correspondências entre pontos-chave que ocorreram entre uma imagem de teste (à esquerda) e uma imagem de treinamento (à direita). Seis correspondências foram estabelecidas entre as duas imagens, as quais "concordam" acerca de um mapeamento entre a imagem de teste e a imagem de treinamento; assim, o método considera que um reconhecimento bem-sucedido aconteceu. Dessa forma, o método retorna a imagem de treinamento com a qual as correspondências foram estabelecidas. O método também retorna a pose do objeto encontrado, que é representada

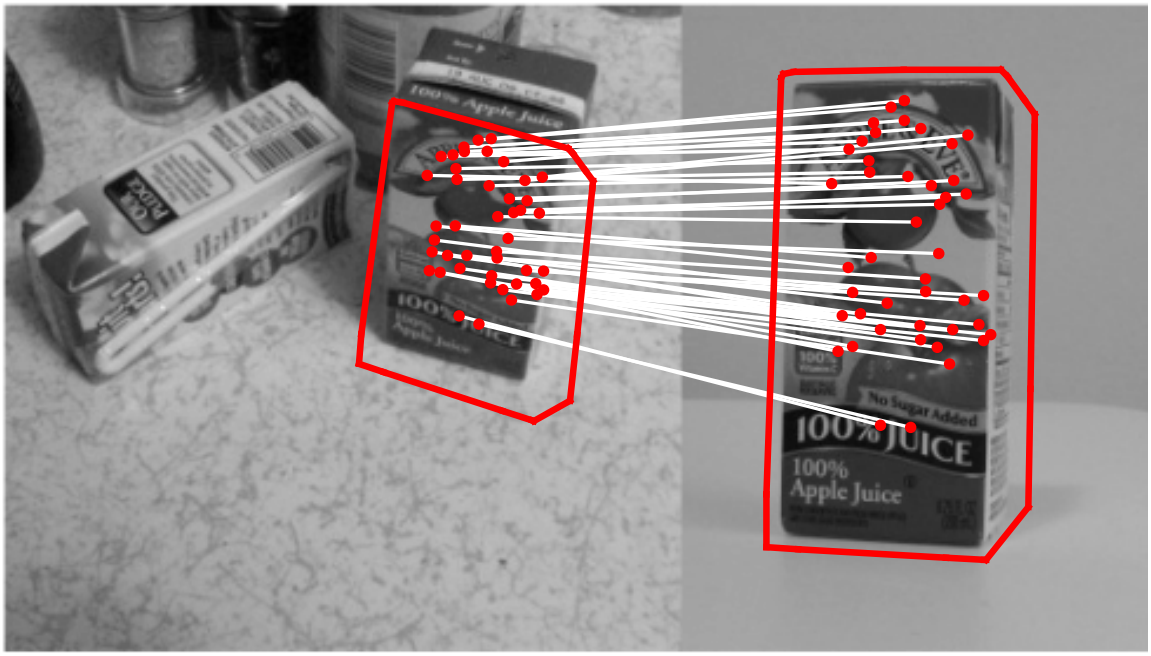


**Figura 2.1:** *Correspondências entre pontos-chave e a pose do objeto encontrada.*

pelo mapeamento entre as imagens (isto é, uma função que mapeia cada posição $x, y$ de uma imagem na posição correspondente na outra imagem). Neste trabalho, são utilizadas transformações afim para mapear imagens; uma transformação afim consiste em uma transformação linear adicionada a uma translação (transformações afim são adequadas para mapear imagens de objetos aproximadamente convexos, e, localmente, possibilitam uma boa aproximação da projeção de um objeto plano).

A ideia de empregar características visuais locais para estabelecer correspondências entre imagens foi inicialmente proposta por Moravec (1981). Os autores propuseram um método para detectar pontos-chave localizados em regiões nas quais a intensidade da imagem varia abruptamente em ambas as direções, tais como nos cantos dos objetos. Entretanto, o método de Moravec (1981) assume que há uma disparidade pequena entre o par de imagens para o qual as correspondências são calculadas. Já Zhang *et al.* (1995) considerou o caso em que uma grande disparidade entre as duas imagens sob comparação pode ocorrer. Os autores exploraram o conceito de descritores locais, utilizando uma janela ao redor de cada ponto-chave a fim de estabelecer correspondências com base na correlação dos *pixels* nessas janelas.

O trabalho seminal de Schmid e Mohr (1997) introduziu a ideia de usar correspondências entre pontos-chave a fim de realizar o reconhecimento de objetos. Os autores utilizaram uma base de dados com 1000 imagens de treinamento, as quais continham os objetos a serem encontrados. Um descritor de pontos-chave mais elaborado do que a simples correlação de *pixels* foi empregado, o qual calcula o vetor de características locais com uso de derivadas da intensidade da imagem. Então, os descritores dos pontos-chave da imagem de teste podiam ser comparados aos descritores dos pontos-chave das imagens de treinamento, o que pode ser feito de forma eficiente mediante o uso de uma árvore para indexar os descritores de treinamento.

O método de Zhang *et al.* (1995) utiliza, como entrada, um par de imagens, e então verifica se

ocorrem correspondências entre os pontos-chave dessas imagens. Já o método de reconhecimento de objetos de Schmid e Mohr (1997) é conceitualmente diferente: ele utiliza, como entrada, uma imagem de teste e um *conjunto* de imagens de treinamento. Entretanto, apesar dessa diferença no tipo de problema considerado – casar um par de imagens versus casar uma imagem com um conjunto de imagens –, ambos os trabalhos empregam a ideia de que correspondências são estabelecidas entre pontos-chave individuais.

Infelizmente, o emprego de correspondências entre pontos-chave individuais apresenta uma desvantagem: limitar a exploração de *vizinhanças* de pontos-chave. Uma vizinhança de pontos-chave decorre do posicionamento relativo de pontos-chave próximos entre si em uma imagem.

A fim de melhor explorar o conceito de vizinhanças de pontos-chave, uma ideia natural é empregar correspondências entre estruturas locais de pontos-chave, em oposição ao uso de correspondências entre pontos-chave individuais.

O tipo mais simples de estrutura de pontos-chave que pode ser empregada para estabelecer correspondências é um par de pontos-chave, de forma que um par de pontos-chave $(p, q)$ em uma imagem seja ligado a um par de pontos-chave $(p', q')$ na outra imagem. Uma correspondência entre pares de pontos-chave contém, intrinsecamente, mais informação do que uma correspondência entre pontos-chave individuais; por exemplo, a distância entre os dois pontos, na imagem. Já uma correspondência entre triplas de pontos-chave – isto é, uma tripla de pontos-chave $(p, q, r)$ em uma das imagens ligada à tripla $(p', q', r')$ na outra – possui, intrisecamente, mais informação do que uma correspondência entre pares de pontos-chave.

Quando correspondências entre estruturas de pontos-chave são empregadas, em vez de correspondências simples entre pontos-chave individuais, o tipo de informação de vizinhança local mais óbvio que passa a ser utilizado é o requisito de co-ocorrência. Por exemplo, se correspondências entre pares são empregadas, para que o par de pontos-chave $(p, q)$ na imagem $I$ estabeleça uma correspondência com o par de pontos-chave $(p', q')$ na imagem $I'$, é necessário que *ambas* as correspondências entre pontos-chave individuais, $(p, p')$ e $(q, q')$, ocorram (entre as imagens $I$ e $I'$). O benefício desse estratégia existe nos casos em que há uma alta probabilidade de que ambas as correspondências sejam corretas simultaneamente; isto é, se somente uma das duas correspondências ocorrer, e a outra não, é um indicativo de que a correspondência que ocorreu, na verdade, possui uma alta probabilidade de ser falsa, e, portanto, pode ser eliminada.

Além de possibilitar o uso do requisito de co-ocorrência, uma outra vantagem do emprego de correspondências entre estruturas de pontos-chave é possibilitar que correspondências candidadas sejam eliminadas mediante a análise de propriedades estruturais presentes em cada uma das duas estruturas de pontos-chave. O seguinte exemplo ilustra o emprego de informações estruturais presentes em triplas de pontos-chave. Considere que uma tripla de pontos-chave forme um triângulo $\mathsf{ABC}$ na imagem de teste e estabeleça uma correspondência com a tripla de pontos-chave que forma o triângulo $\mathsf{A'B'C'}$ na imagem de treinamento, em que as correspondências foram estabelecidas entre os vértices $\mathsf{A}$ e $\mathsf{A'}$, entre $\mathsf{B}$ e $\mathsf{B'}$ e entre $\mathsf{C}$ e $\mathsf{C'}$. Assim, se a aresta $\mathsf{AB}$ for maior do que a aresta $\mathsf{BC}$ enquanto a aresta $\mathsf{A'B'}$ é muito menor do que a aresta $\mathsf{B'C'}$, então os pontos-chave não variaram de forma consistente entre as duas imagens, de forma que essa correspondência entre triplas pode ser eliminada devido à alta probabilidade de que ao menos uma das três correspondências entre pontos-chave (vértices) seja falsa. Entretanto, se, em vez de triplas de pontos-chave, fossem utilizadas correspondências somente entre pares de pontos-chave, não seria possível fazer uma veri-

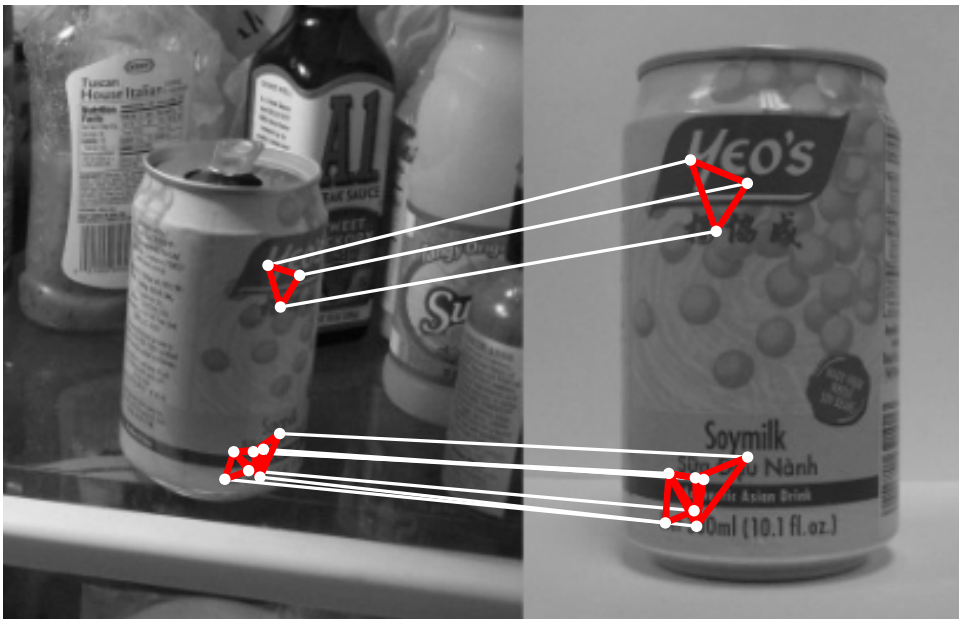


**Figura 2.2:** *Correspondências estabelecidas entre triplas de pontos-chave.*

ficação similar acerca da consistência nas distâncias dos pontos-chave na imagem, já que há somente uma informação de distância entre um par de pontos-chave (isto é, não existe utilidade em comparar somente as distâncias $\mathsf{AB}$ e $\mathsf{A'B'}$).

A tese fundamental deste trabalho é a de que o emprego de correspondências entre triplas de pontos-chave oferece vantagens em comparação ao emprego de correspondências entre pontos-chave individuais ou ao emprego de correspondências entre pares de pontos-chave. O uso de triplas de pontos-chave possibilita que informações estruturais mais ricas sejam utilizadas a fim de estabelecer correspondências, o que acarreta na geração de um conjunto de correspondências de melhor qualidade, com uma maior proporção (e quantidade) de correspondências corretas.

A Figura 2.2 ilustra as correspondências entre grafos-chave que foram geradas entre uma imagem de teste (à esquerda) e uma imagem de treinamento (à direita), com uso do método proposto neste trabalho.

## 2.2 Correspondências entre pontos-chave individuais

A maioria dos métodos que empregam pontos-chave para realizar o reconhecimento de objetos segue a estratégia de estabelecer correspondências entre pontos-chave individuais. Uma correspondência individual, $(p, p')$, liga um ponto-chave $p$ na imagem de teste a um ponto-chave $p'$ em uma imagem de treinamento.

O estabelecimento de correspondências entre pontos-chave usualmente envolve uma etapa inicial que identifica correspondências candidatas com base na similaridade entre os descritores dos pontos-chave. A fim de estabelecer essas correspondências iniciais de uma forma eficiente, uma árvore é utilizada para indexar os pontos-chave das imagens de treinamento de acordo com seus descritores locais. Assim, um descritor de um ponto-chave da imagem de teste somente precisa ser comparado a uma pequena fração do total de descritores indexados, pois a indexação permite que somente comparações relevantes sejam feitas (isto é, comparações entre pontos-chave que apresentem uma

alta probabilidade de constituírem uma correspondência correta).

Após as correspondências iniciais entre pontos-chave serem estabelecidas com uso da árvore indexadora e da similaridade entre descritores, uma possibilidade é empregar *todas* essas correspondências na etapa seguinte, em que a pose do objeto é estimada. Uma outra possibilidade é utilizar alguma estratégia para eliminar parte dessas correspondências, mediante o uso de informações estruturais de vizinhanças de pontos-chave para excluir correspodências que apresentem uma alta probabilidade de serem falsas; então, na etapa seguinte, em que a pose do objeto é estimada, somente as correspodências que não foram eliminadas são utilizadas.

### 2.2.1 Correspondências estabelecidas somente pela semelhança de descritores

A estratégia mais simples para estabelecer correspondências entre pontos-chave individuais envolve o uso apenas da semelhança entre os descritores dos pontos-chave. Abaixo, são mencionados alguns trabalhos relevantes nesse contexto.

#### Lowe (2004)

Lowe (2004) propõe uma estratégia para lidar com correspondências entre pontos-chave individuais que são estabelecidas somente com uso da similaridade entre os descritores. O autor também propõe um método para detectar pontos-chave e calcular seus descritores locais.

**Pontos-chave SIFT.** A principal contribuição de Lowe (2004) é um método detector de pontos-chave denominado SIFT (*Scale Invariant Feature Transform*).

Características locais SIFT são amplamente empregadas em visão computacional, devido ao seu bom nível de invariância em resposta a transformações visuais causadas por mudanças de iluminação, de escala e de ângulo de visão. O descritor SIFT, um vetor de características com 128 dimensões, consiste em um histograma de gradientes da imagem na região local considerada.

Esta tese emprega pontos-chave SIFT a fim de representar as informações visuais locais em cada imagem, embora utilize uma estratégia de estabelecimento de correspondências mais elaborada do que aquela proposta originalmente em Lowe (2004).

**Estratégia para estabelecer correspondências.** Na estratégia proposta por Lowe (2004) estabelecer correspondências entre pontos-chave individuais, cada ponto-chave $p$ da imagem de teste pode dar origem a somente *uma* correspondência, $(p, p')$, a qual ocorre com o ponto-chave $p'$ (de uma imagem de treinamento) tal que o descritor de $p'$ é o mais semelhante ao descritor de $p$; isto é, o descritor de $p'$ é o vizinho mais próximo do descritor de $p$ no espaço de descritores.

A fim de estabelecer correspondências entre pontos-chave de uma forma eficiente, deve ser empregada uma árvore para indexar os descritores dos pontos-chave das imagens de treinamento.

Após as correspondências serem inicialmente estabelecidas com uso da árvore indexadora, uma etapa posterior de eliminação de correspondências é utilizada, a qual usa a distância euclidiana $d_1$ entre o descritor de $p$ e seu vizinho mais próximo, e também usa a distância euclidiana $d_2$ entre o descritor de $p$ e seu segundo vizinho mais próximo. Se o valor de $d_2$ for maior do que 80% do valor de $d_1$, significa que há pouca confiança na correspondência entre $p$ e seu vizinho mais próximo, de forma que essa correspondência é eliminada. Entretanto, essa etapa de eliminação de correspondências emprega somente a distânca entre os descritores dos pontos-chave, sem analisar as informações existentes em vizinhanças de pontos-chave.

### Gordon e Lowe (2006)

O método proposto por Gordon e Lowe (2006) utiliza pontos-chave SIFT e a estratégia de estabelecimento de correspondências apresentada em Lowe (2004). Entretanto, o problema considerado por Gordon e Lowe (2006) é mais complexo do que o casamento de imagens. Os autores propõem um método de reconhecimento de objetos em que cada objeto de treinamento é representado por um modelo tridimensional.

**Correspondências entre uma imagem e um modelo de objeto 3D.** Um modelo tridimensional de um objeto é calculado com uso de imagens de treinamento obtidas ao redor do objeto. Pontos-chave SIFT são detectados e triangulações são realizadas a fim de casar esses pontos, produzindo, como resultado, um modelo composto de alguns milhares de pontos 3D. Cada ponto 3D de um modelo está associado a diferentes pontos-chave SIFT, os quais consistem no "mesmo" ponto-chave detectado em diferentes imagens de treinamento.

No método de Gordon e Lowe (2006), após os modelos tridimensionais dos objetos de treinamento serem criados, o reconhecimento de objetos pode ser realizado. Pontos-chave SIFT são extraídos de uma imagem de teste, e correspondências são estabelecidas com os pontos-chave SIFT dos objetos de treinamento, em que a estratégia proposta por Lowe (2004) é utilizada – isto é, rejeitar correspodências nas quais a distância ao descritor do segundo vizinho mais próximo é maior do que 80% da distância ao descritor do primeiro vizinho mais próximo.

**Estimação de pose com uso do algoritmo RANSAC.** Após as correspondências entre pontos-chave estarem definidas, ocorre a etapa de estimação de pose dos objetos. Essa etapa utiliza as correspondências estabelecidas entre os pontos 2D (na imagem de treinamento) e 3D (nos modelos) a fim de encontrar uma pose com uso do algoritmo de estimação de poses RANSAC (*Random Sample Consensus*), proposto por Fischler e Bolles (1981).

A partir do conjunto de correspondências entre pontos-chave 2D-3D, o algoritmo de estimação de pose RANSAC, iterativamente, seleciona três correspondências (de forma aleatória) e as emprega para avaliar uma pose candidate; há necessidade de um mínimo de três correspondências entre pontos para que seja possível calcular uma instanciação de um modelo 3D em uma imagem 2D.

A cada iteração, uma pose candidata diferente é avaliada; várias iterações são empregadas a fim de analisar um grande número de poses candidatas. A confiança associada a uma pose candidata é dada pelo total de correspondências que "concordam" com a instanciação da imagem no modelo 3D.

### Hsiao *et al.* (2010)

O método proposto por Hsiao *et al.* (2010) também utiliza modelos tridimensionais para realizar o reconhecimento de objetos, semelhante ao trabalho de Gordon e Lowe (2006).

Uma modificação de Hsiao *et al.* (2010) em relação a Gordon e Lowe (2006) é que o método de Hsiao *et al.* (2010) obtém várias distorções artificiais de cada imagem de treinamento, as quais são tratadas como imagens de treinamento adicionais. Isso gera um maior conjunto de pontos-chave, que enriquecem o modelo 3D do objeto.

Uma outra contribuição de Hsiao *et al.* (2010) foi disponibilizar uma base de dados que pode ser usada para avaliar métodos de reconhecimento de objetos. Essa base de dados é composta de 500 imagens de teste e 250 imagens de treinamento (com dez tipos de objetos diferentes). Neste

trabalho, a base de dados proposta por Hsiao *et al.* (2010) é utilizada por a fim de avaliar o método proposto e realizar as comparações experimentais com os outros métodos analisados.

### Philbin *et al.* (2007)

O trabalho de Philbin *et al.* (2007) apresenta um método de recuperação de imagens. A tarefa de recuperação de imagens, embora seja semelhante à de reconhecimento de objetos, apresenta diferenças: recuperação de imagens envolve bases de dados com um número muito mais elevado de imagens de treinamento, na ordem de milhões.

**Recuperação de imagens.** A diferença essencial entre métodos de recuperação de imagens e métodos de reconhecimento de objetos é que, enquanto métodos de reconhecimento de objetos explicitamente calculam a distância euclidiana entre os descritores dos pontos-chave (na etapa inicial que emprega a árvore de indexação), métodos de recuperação de imagens somente calculam a distância euclidiana entre descritores da imagem de teste e *aglomerados* (*clusters*) de descritores de imagens de treinamento. Assim, correspondências são estabelecidas entre pontos-chave da imagem de teste e conjuntos de pontos-chave de imagens de treinamento; isso possibilita estimar, de forma eficiente, a semelhança entre imagens, para enfim retornar um pequeno conjunto de imagens de treinamento, as quais apresentaram a maior similaridade à imagem de teste.

Uma desvantagem de métodos de recuperação de imagens, em relação a métodos de reconhecimento de objetos, é que a estratégia de estabelecer correspondências com aglomerados de pontos-chave de treinamento (em vez de estabelecer correspondências com pontos-chave individuais) acarreta numa menor proporção de correspondências entre pontos-chave corretas, devido à perda de precisão causada pelo uso de aglomerados de descritores. Dessa forma, métodos de recuperação de imagens apresentam um desempenho inferior a métodos de reconhecimento de objetos, embora sejam capazes de lidar com bases de imagens maiores.

**Pontos-chave invariantes a transformações afim.** O método de recuperação de imagens de Philbin *et al.* (2007) empregou pontos-chave que apresentam invariância não somente a transformações de escala da imagem, mas também a transformações afim locais (esse método detector de pontos-chave foi proposto por Mikolajczyk e Schmid (2005) ).

O uso de correspondências entre pontos-chave invariantes a transformações afim, como empregado por Philbin *et al.* (2007), possibilita que *uma única* correspondência entre pontos-chave contenha informação suficiente para que uma pose candidata seja estabelecida. Já no método proposto neste trabalho, uma correspondência entre grafos-chave (isto é, entre triplas de pontos-chave) é suficiente para instanciar uma pose candidata. Nesse contexto, há três vantagens em utilizar correspondências entre grafos-chave. *Primeiro*, uma pose candidata gerada por uma correspondência entre grafos-chave é mais robusta que uma pose candidata gerada somente por uma correspondência entre pontos-chave invariantes a transformações afim, já que, no caso dos grafos-chave, três regiões da imagem (pontos-chave) são utilizadas para calcular a pose candidata, em oposição ao uso de somente uma região, no caso do método de Philbin *et al.* (2007). *Segundo*, correspondências entre grafos-chave possibilitam que informações estruturais sejam empregadas a fim de eliminar correspondências com uma alta probabiliade de serem falsas, o que melhora tanto o desempenho quanto a eficiência do reconhecimento de objetos. *Terceiro*, a detecção de pontos-chave invariantes a transformações afim é computacionalmente muito mais cara do que a detecção de pontos-chave

invariantes somente a mudanças de escala (como, por exemplo, pontos-chave SIFT), o que limita o emprego de pontos-chave invariantes a transformações afim em aplicações tais como robótica.

### 2.2.2 Uso de informações estruturais para eliminar correspondências

A etapa que estabelece correspondências entre pontos-chave com uso da árvore indexadora emprega somente a similaridade entre os descritores. Entretanto, entre essas correspondências inicialmente estabelecidas, há uma grande proporção de correspondências falsas. À medida que o número de correspondências falsas aumenta, há um aumento drástico no número de poses de objeto candidatas que devem ser avaliadas na etapa final de estimação de pose, o que eleva o custo computacional e pode diminuir o desempenho.

Uma possibilidade para eliminar algumas correspondências que apresentem uma alta probabilidade de serem falsas envolve empregar informações estruturais de vizinhanças de pontos-chave. A ideia é que o estabelecimento de uma correspondência $(p, p')$ entre pontos-chave, entre a imagem $I$ e $I'$, não dependa somente dos pontos-chave $p$ e $p'$ mas também dependa de outras correspondências entre pontos-chave existentes entre as imagens $I$ e $I'$.

#### Sattler *et al.* (2009)

Um exemplo de método que usa correspondências entre pontos-chave individuais e emprega uma etapa posterior para eliminar correspondências utilizando informações de vizinhanças de pontos-chave é SCRAMSAC (*Spatial Consensus Random Sample Consensus*), proposto por Sattler *et al.* (2009). Inicialmente, SCRAMSAC obtém correspondências entre pontos-chave de forma similar a Lowe (2004), com uso somente da semelhança entre descritores de pontos-chave. Então, numa etapa posterior, SCRAMSAC explora o conceito de vizinhanças de pontos-chave, a fim de remover correspondências que provavelmente são falsas.

O princípio que é explorado por SCRAMSAC é o fato de que uma correspondência entre pontos-chave correta normalmente ocorre próxima, na imagem, a outras correspondências entre pontos-chave corretas (esse princípio também é explorado pelo método dos grafos-chave proposto nesta tese, já que cada grafo-chave na imagem de teste consiste em uma *vizinhança local* de três pontos-chave).

Dada uma correspondência entre pontos-chave $(p, p')$, SCRAMSAC elimina essa correspondência caso não existam outras correspondências ligando um ponto-chave próximo a $p$ (na imagem de teste) a um ponto-chave próximo a $p'$ (na imagem de treinamento). Especificamente, para que a correspondência $(p, p')$ seja considerada válida, é necessário que um mínimo de 55% dos pontos-chave que estão numa vizinhança circular ao redor de $p$ tenham estabelecido correspondência com algum ponto-chave numa vizinhança circular ao redor de $p'$.

#### Jégou *et al.* (2010)

Jégou *et al.* (2010) propuseram um método de recuperação de imagens que explora, de uma forma simples, vizinhanças de pontos-chave, a fim de melhorar o desempenho.

**Variações de escala e de orientação de pontos-chave.** Cada ponto-chave SIFT possui três atributos associados a ele: uma *posição* na imagem, isto é, uma coordenada bidimensional $x, y$; um valor de *escala*, relacionado ao tamanho da área na imagem na qual o descritor SIFT é calculado;

e um valor de *orientação*, que consiste em um ângulo associado ao ponto-chave (ângulo em relação ao eixo horizontal da imagem e em sentido anti-horário).

Para cada correspondência entre pontos-chave $(p, p')$, é possível calcular a variação na escala e a variação na orientação que ocorrem entre a imagem de treinamento e a imagem de teste.

Como um exemplo de variação na escala dos pontos-chave, seja $(p, p')$ uma correspondência entre pontos-chave, em que $p$ pertence à imagem $I$ e $p'$ pertence à imagem $I'$. Se a escala de $p$ é 1.5 e a escala de $p'$ é 3.0, ocorreu uma variação na escala de $2\times$ entre a imagem de teste e a imagem de treinamento. Assumindo que a correspondência $(p, p')$ seja correta, então o objeto aparece, na imagem de teste, com, aproximadamente, metade do tamanho em relação ao objeto na imagem de treinamento; isto é, todas as outras correspondências entre pontos-chave que também são corretas precisam apresentar uma variação na escala que também seja, aproximadamente, de $2\times$. Ou seja, correspondências cuja variação na escala dos pontos-chave seja muito diferente de $2\times$ não são consistentes, e, portanto, são falsas.

Como um exemplo de mudança na orientação dos pontos-chave, seja $(p, p')$ uma correspondência entre pontos-chave (entre as imagens $I$ e $I'$). Se a orientação de $p$ é $30°$ e a orientação de $p'$ é $45°$, então houve uma variação de $15°$, no sentido anti-horário, entre a imagem de teste e a imagem de treinamento. Assumindo que a correspondência $(p, p')$ seja correta, então o objeto aparece, na imagem de teste, rotacionado, aproximadamente, em $15°$ no sentido anti-horário, em relação ao objeto na imagem de treinamento; isto é, todas as outras correspondências entre pontos-chave que também são corretas precisam apresentar uma variação na orientação que também seja, aproximadamente, de $15°$ no sentido anti-horário. Ou seja, correspondências cuja variação na orientação dos pontos-chave foi muito diferente de $15°$ no sentido anti-horário não são consistentes, e, portanto, são falsas.

**Verificação da consistência nas variações de escala e de orientação.** O método proposto por Jégou *et al.* (2010) verifica se as variações na escala dos pontos-chave SIFT são consistentes entre si. A existência de muitas correspondências com uma variação semelhante na escala dos pontos-chave acarreta numa similaridade maior entre o par de imagens (de teste e de treinamento), de forma que há um aumento na probabilidade de que essa imagem de treinamento seja retornada pelo método de recuperação de imagens.

De forma semelhante, o método também verifica se as variações na orientação dos pontos-chave SIFT são consistentes entre si. A existência de muitas correspondências com uma variação semelhante na orientação dos pontos-chave acarreta numa similaridade maior entre o par de imagens (de teste e de treinamento), de forma que há um aumento na probabilidade de que essa imagem de treinamento seja retornada pelo método de recuperação de imagens.

## 2.3 Correspondências entre pares de pontos-chave

Estabelecer correspondências entre pares de pontos-chave é uma forma mais elaborada de utilizar informações de vizinhanças de pontos-chave, em comparação ao uso de simples correspondências entre pontos-chave individuais.

No caso de correspondências entre pontos-chave individuais serem empregadas, há uma limitação natural no tipo de informação de vizinhança de pontos-chave que pode ser explorada. Por exemplo, SCRAMSAC, a fim de verificar se uma correspondência entre pontos-chave $p$ e $p'$ é correta, faz meramente uma contagem da quantidade de correspondências que ocorrem próximas a ambos $p$

e $p'$; isto é, SCRAMSAC não analisa, de fato, a *estrutura* da vizinhança de pontos-chave, a qual decorre do posicionamento relativo dos pontos-chave na imagem. Já o método de recuperação de imagens proposto por Jégou *et al.* (2010) nem sequer utiliza "vizinhanças locais" de pontos-chave, pois realiza meramente uma verificação global acerca da consistência nas variações de escala e de orientação dos pontos-chave.

Embora o emprego de correspondências entre pares de pontos-chave não possibilite uma análise aprofundada da estrutura de vizinhanças de pontos-chave (em comparação ao uso de correspondências entre triplas de pontos-chave), utilizar correspondências entre pares de pontos-chave possibilita uma melhor exploração das vizinhanças em comparação ao uso de correspondências entre pontos-chave individuais.

Se uma correspondência entre os pares de pontos-chave $(p, q)$ e $(p', q')$ é estabelecida, entre as imagens $I$ e $I'$, uma informação possível de ser calculada é a distância entre $p$ e $q$ (distância em *pixels* na imagem $I$) e a distância entre $p'$ e $q'$ (distância em *pixels* na imagem $I'$). Uma mudança de escala entre as duas imagens afeta de uma forma similar a distância entre o par de pontos-chave e a escala individual de cada ponto-chave. Por exemplo, se o objeto aparece, na imagem de teste, com metade do tamanho em que aparece na imagem de treinamento, então tanto a escala dos pontos-chave é reduzida pela metade quanto a distância entre o par de pontos-chave na imagem também é reduzida pela metade. Assim, é possível verificar se há consistência na correspondência entre o par de pontos-chave mediante a análise das variações desses valores de escala e de distância entre pontos-chave na imagem. O método de reconhecimento de objetos apresentado em Hao *et al.* (2013) utiliza uma ideia semelhante a essa.

Com o uso de correspondências entre pares de pontos-chave, também é possível verificar se há consistência entre as variações de orientação dos pontos-chave e de suas posições na imagem. Uma linha reta que conecta um par de pontos-chave em uma imagem possui, intrinsecamente, uma orientação (ângulo), medida em relação ao eixo horizontal da imagem. Uma rotação que ocorra entre duas imagens afeta de uma forma similar a orientação dessa linha reta entre o par de pontos-chave e também a orientação de cada ponto-chave individual. Por exemplo, se o objeto aparece, na imagem de teste, rotacionado em 30° (no sentido anti-horário) em relação à sua orientação na imagem de treinamento, então tanto a orientação dos pontos-chave é rotacionada em 30° quanto a orientação da linha reta conectando o par de pontos-chave na imagem também é rotacionada em 30°. Assim, é possível verificar se há consistência na correspondência entre o par de pontos-chave mediante a análise das variações dessas orientações. O método proposto nesta tese realiza uma verificação semelhante a essa, acerca da consistência nas orientações de pontos-chave e de linhas retas conectando pontos-chave em uma imagem, embora utilize correspondências entre triplas de pontos-chave, em vez de correspondências entre pares de pontos-chave.

O método apresentado em Hao *et al.* (2013) utiliza correspondências entre pares de pontos-chave para realizar o reconhecimento de objetos com uso de um modelo tridimensional para cada objeto de treinamento (de forma semelhante ao método de Gordon e Lowe (2006)). No método proposto por Hao *et al.* (2013), inicialmente, correspondências são estabelecidas entre pontos-chave individuais, com o emprego de uma estratégia semelhante à do método SCRAMSAC a fim de eliminar correspondências com uso de informações de vizinhanças de pontos-chave. Em seguida, as correspondências entre pontos-chave individuais são transformadas em correspondências entre pares de pontos-chave, o que permite ao método verificar a consistência entre a distância entre o par de

pontos-chave na imagem de teste (em *pixels*) e a distância entre o par de pontos 3D no modelo tridimensional do objeto (na unidade de distância do modelo); além disso, o método também verifica a consistência existente entre a escala dos pontos-chave na imagem de teste e a escala dos pontos 3D no modelo (em que a escala de um ponto 3D é calculada como a média das escalas dos ponto-chave 2D que estão associados ao ponto 3D). Entretanto, em comparação ao método apresentado nesta tese, o método de Hao *et al.* (2013) não verifica se há consistência na variação das orientações dos pontos-chave e na variação da orientação de linhas retas conectando pares de pontos-chave na imagem de teste.

## 2.4 Correspondências entre triplas de pontos-chave

Neste trabalho, em vez de utilizar pontos-chave individuais ou pares de pontos-chave, correspondências são estabelecidas entre *triplas* de pontos-chave. Isto é, uma tripla de pontos-chave na imagem de teste, $(p, q, r)$, estabelece uma correspondência com uma tripla de pontos-chave em uma imagem de treinamento, $(p', q', r')$.

### 2.4.1 Informações estruturais em triplas de pontos-chave

O uso de triplas de pontos-chave apresenta vantagens em relação ao uso de pares de pontos-chave, pois as informações estruturais em uma vizinhança de três pontos são mais ricas do que aquelas existentes em uma vizinhança de somente dois pontos. Especificamente, há dois tipos de informações que triplas de pontos-chave possuem mas pares de pontos-chave não possuem.

O primeiro tipo de informação que é encontrada em uma tripla de pontos-chave, mas não em um par de pontos-chave, é o conceito topológico de sentido de ciclo. Há dois sentidos de ciclo possíveis: horário ou anti-horário. Já que uma transformação visual de espelhamento não é possível de ocorrer entre a imagem de teste e a imagem de treinamento, então ambas as triplas de pontos-chave envolvidas em uma correspondência devem apresentar o mesmo sentido de ciclo.

O segundo tipo de informação que é encontrada em uma tripla de pontos-chave, mas não em um par de pontos-chave, são três linhas retas conectando pares de pontos-chave na imagem; isso possibilita calcular a distância, na imagem, entre cada um desses pares de pontos-chave. Por exemplo, na tripla $(p, q, r)$, há uma linha reta que liga $p$ a $q$, outra que liga $q$ a $r$ e outra que liga $r$ a $p$ (em comparação, um par de pontos-chave $(p, q)$ possui somente uma linha reta, conectando $p$ e $q$). Quando ocorre uma mudança de escala entre duas imagens, o comprimento de cada uma dessas linhas retas variará aproximadamente na mesma proporção; por exemplo, se o objeto aparecer, na imagem de teste, com metade do tamanho que possui na imagem de treinamento, então cada uma das três linhas retas aparecerá, na imagem de teste, com, aproximadamente, metade do tamanho que possui na imagem de treinamento. Assim, é possível verificar se há consistência na variação do tamanho dessas três linhas retas, e, caso não haja consistência, a correspondência entre triplas pode ser eliminada.

### 2.4.2 Trabalhos anteriores que utilizaram triplas de pontos-chave

O uso de correspondências entre triplas de pontos-chave já foi investigado em trabalhos anteriores na literatura; por exemplo, Hao *et al.* (2012), Zitnick *et al.* (2007) e Morimitsu *et al.* (2011). Entretanto, há uma diferença fundamental entre esses trabalhos anteriores na literatura e o

método proposto nesta tese para utilizar triplas de pontos-chave: os métodos anteriores na literatura seguiram a estratégia de calcular, em cada imagem de *treinamento*, uma quantidade representativa de triplas de pontos-chave, as quais são, então, explicitamente armazenadas a fim de serem usadas em tempo de aplicação. Dessa forma, o reconhecimento de objetos ocorre via o estabelecimento de correspondências entre essas triplas (que estão pré-calculadas e armazenadas) e os pontos-chave extraídos da imagem de teste. Já o método proposto nesta tese calcula as triplas de pontos-chave na imagem de *teste*, mediante o uso de um algoritmo eficiente de seleção de triplas.

Uma desvantagem da estratégia de pré-calcular triplas de pontos-chave em imagens de treinamento e armazená-las é que a quantidade de triplas possíveis cresce muito mais rapidamente do que a quantidade de pontos-chave. Por exemplo, uma imagem com alguns milhares de pontos-chave provavelmente geraria centenas de milhares de triplas. Assim, uma grande quantidade de memória seria necessária para armazenar todas essas triplas de pontos-chave, o que limita a quantidade de imagens de treinamento que podem ser utilizadas.

Em vez de armazenar triplas de pontos-chave calculadas em imagens de treinamento, esta tese emprega um algoritmo para selecionar, de forma eficiente, triplas de pontos-chave a partir da imagem de teste; já que as triplas são obtidas na imagem de teste e imediatamente utilizadas, não há necessidade de armazenar triplas de pontos-chave para uso futuro. Somente os *pontos-chave* das imagens de treinamento devem ser armazenados, e seus descritores, indexados. Assim, correspondências são, inicialmente, estabelecidas entre as triplas de pontos-chave na imagem de teste e os pontos-chave nas imagens de treinamento, o que resulta, então, em correspondências entre triplas de pontos-chave.

### Hao *et al.* (2012)

Hao *et al.* (2012) consideram o problema de reconhecimento de objetos em que cada objeto de treinamento é representado como um modelo tridimensional. Uma diferença do método em Hao *et al.* (2012) em comparação aos trabalhos de Gordon e Lowe (2006) e Hsiao *et al.* (2010), os quais também empregam modelos tridimensionais de objetos, é que o método de Hao *et al.* (2012) emprega um número muito maior de imagens de treinamento a fim de calcular o modelo 3D de cada objeto.

As triplas de pontos-chave de treinamento são calculadas a partir do modelo 3D do objeto, e então são armazenadas a fim de serem empregadas em tempo de aplicação. Cada tripla é do formato $(p, q, r)$, em que $p$, $q$ e $r$ são pontos 3D no modelo (cada ponto 3D estando associado ao "mesmo" pontos-chave SIFT em diferentes imagens de treinamento).

A partir de cada modelo tridimensional de objeto, muitas triplas de pontos 3D são calculadas. Algumas dessas triplas são compostas por pontos 3D que estão próximos entre si no modelo, enquanto outras triplas são compostas por pontos 3D que estão mais distantes entre si no modelo. Isso possibilita que os objetos possam aparecer em diferentes tamanhos na imagem de teste: se o objeto aparecer na imagem de teste com um tamanho pequeno, isto é, sendo observado a partir de uma longa distância, então as triplas utilizadas seriam aquelas compostas por pontos 3D distantes entre si no modelo; já se o objeto aparecer na imagem de teste sendo observado a partir de uma curta distância, as triplas utilizadas seriam aquelas compostas de pontos 3D próximos entre si no modelo do objeto. Entretanto, essa abordagem multi-escala de geração de triplas de pontos 3D aumenta, ainda mais, a quantidade total de triplas que devem ser armazenadas.

O método proposto por Hao *et al.* (2012) realiza a verificação da consistência existente nas variações de pontos-chave e pontos 3D envolvidos em uma correspondência entre triplas. Os autores verificam a consistência existente entre: a distância entre os pontos-chave na imagem de teste; a distância entre os pontos-chave em imagens de treinamento nas quais esses pontos-chave aparecem (com uso de uma média das distâncias); a escala dos pontos-chave SIFT na imagem de teste; e a escala dos pontos-chave SIFT em imagens de treinamento nas quais esses pontos-chave apareçam (com uso de uma média das escalas).

O método de Hao *et al.* (2012) utiliza o conceito topológico de sentido de ciclo das triplas de pontos-chave, o qual pode ser horário ou anti-horário. Com isso, o método elimina correspondências entre um par de triplas em que uma das triplas está em um dos sentidos e a outra tripla está no outro sentido.

Assim, mediante a verificação da consistência existente nas variações de escala e de distância entre pontos-chave chave na imagem, e também com a verificação do sentido de ciclo das triplas de pontos-chave, o método de Hao *et al.* (2012) é capaz de eliminar correspondências entre triplas que apresentem uma alta probabilidade de serem falsas. Contudo, em comparação ao método apresentado nesta tese, o método de Hao *et al.* (2013) não verifica a consistência na variação das orientações dos pontos-chave e na variação da orientação de linhas retas conectando pares de pontos-chave na imagem de teste. Assim, o método de Hao *et al.* (2013) é menos eficiente na eliminação de correspondências falsas, em comparação ao método apresentado nesta tese.

### Hashimoto e Cesar Jr (2009) e Morimitsu *et al.* (2011)

O conceito de empregar correspondências entre triplas de pontos-chave tratadas como grafos de pontos-chave (grafos-chave) foi originalmente introduzido em Hashimoto e Cesar Jr (2009) e posteriormente empregado em Morimitsu *et al.* (2011) em uma tarefa de auto-localização, cujo objeto a ser reconhecido eram placas com textos.

Uma diferença fundamental entre o método de Hashimoto e Cesar Jr (2009) e o método proposto nesta tese é que o trabalho de Hashimoto e Cesar Jr (2009) propõe um método cujo objetivo central é a eficiência no reconhecimento de objetos; de fato, em Morimitsu *et al.* (2011), esse método foi executado em aparelhos celulares *smartphone* comuns. Contudo, somente *uma* imagem de treinamento podia ser utilizada. Já o método proposto nesta tese é escalável, isto é, pode ser empregado, de forma eficiente, com uso de uma grande quantidade de imagens de treinamento.

O método proposto em Morimitsu *et al.* (2011) emprega, como descritores locais de imagens, coeficientes da transformada de Fourier calculada a partir da linha reta conectando um par de pontos-chave em uma imagem (aresta do grafo-chave); isto é, cada aresta de um grafo-chave gera uma característica local distinta. Tal descritor apresenta como vantagem o fato de possuir baixa dimensionalidade, o que diminui o custo computacional na etapa inicial que estabelece correspondências com uso da árvore de indexação de descritores (em comparação a um descritor de alta dimensionalidade, como SIFT, com 128 dimensões). Já o método proposto nesta tese utiliza grafos-chave cujas características locais são pontos-chave SIFT associados aos vértices dos grafos-chave.

Um avanço presente no método proposto nesta tese em relação ao método de Morimitsu *et al.* (2011) é o uso de mais propriedades estruturais para eliminar correspondências candidatas, com emprego da orientação e da escala dos pontos-chave SIFT e da orientação e do comprimento das arestas dos grafos-chave, além do sentido de ciclo das triplas; já o método em Morimitsu *et al.*

(2011) utiliza somente o sentido de ciclo das triplas para eliminar correspondências candidatas.

Outro avanço do método proposto nesta tese é o emprego de diferentes (e complementares) triangulações de Delaunay a fim de gerar grafos-chave na imagem de teste, enquanto o método em Morimitsu *et al.* (2011) emprega somente uma única triangulação de Delaunay para gerar grafos-chave na imagem de teste (e ainda demanda o cálculo de muitos grafos-chave na imagem de treinamento, os quais são armazenados a fim de serem usados em tempo de aplicação).

**Grafos-chave com quatro ou mais pontos-chave.** O método apresentado em Morimitsu *et al.* (2011) propõe uma abordagem geral para representar estruturas locais de pontos-chave como grafos-chave, em que grafos-chave podem ser compostos de três, quatro ou mais pontos-chave. Um grafo-chave é definido como um *circuito* de pontos-chave, em que o único sentido de ciclo válido é assumido como sendo o anti-horário.

Embora triplas de pontos-chave possuam, intrinsecamente, mais informações estruturais do que pares de pontos-chave (o sentido de ciclo da tripla e as três linhas retas conectando pares de pontos-chave), não há um tipo específico de informação estrutural que estruturas com quatro pontos-chave possuam mas que estruturas com três pontos-chave não possuam.

Apesar de grafos-chave com quatro ou mais pontos-chave não possuírem, intrinsecamente, um tipo de informação estrutural que triplas de pontos-chave não tenham, é possível que, em algumas aplicações, o uso de grafos-chave com quatro (ou mais) pontos-chave apresente vantagens. O uso de grafos-chave com um número maior de pontos-chave aumenta a exigência de similaridade entre os grafos-chave da imagem de teste e das imagens de treinamento; por exemplo, se grafos-chave com quatro pontos-chave são utilizados, o requisito de co-ocorrência exigiria que todas as quatro correspondências entre pontos-chave individuais sejam simultaneamente corretas. Essa maior exigência na similaridade entre os grafos-chave causaria um aumento na proporção de correspondências que são eliminadas, o que pode ser útil em um contexto em que muitas correspondências entre pontos-chave individuais são inicialmente estabelecidas na etapa que emprega a árvore de indexação de descritores de pontos-chave de treinamento. Por exemplo, no método de Hao *et al.* (2013), que emprega correspondências entre pares de pontos-chave, os descritores utilizados não são SIFT (com 128 dimensões), mas, sim, descritores DAISY (propostos por Winder *et al.* (2009)), que possuem 32 dimensões; embora isso diminua o custo computacional para estimar a distância euclidiana entre descritores, há perda de precisão devido à menor dimensionalidade, o que exige que um número maior de correspondências sejam inicialmente estabelecidas a fim de aumentar o número de correspondências corretas encontradas. Assim, o emprego de grafos-chave com quatro vértices poderia ser uma estratégia para eliminar um maior número de correspondências inicialmente estabelecidas e que são incorretas. Entretanto, um problema em utilizar grafos-chave com um maior número de pontos-chave é a explosão combinatória, a qual acarreta na necessidade de obter uma quantidade maior de grafos-chave a fim de gerar um conjunto representativa de grafos-chave. Nesse caso, seria necessária uma análise acerca do melhor balanceamento entre o número de correspondências entre pontos-chave inicialmente estabelecidas e a quantidade de grafos-chave extraídos da imagem de teste.

# Capítulo 3

# Metodologia

## 3.1 Visão geral do método de reconhecimento de objetos proposto

No método proposto nesta tese, o resultado de um reconhecimento de objetos bem-sucedido é uma transformação afim que mapeia a imagem de teste e a imagem de treinamento que contém o objeto encontrado. Múltiplos objetos podem ser reconhecidos em uma mesma imagem de teste, cada um associado a uma imagem de treinamento diferente.

Antes que o método de reconhecimento de objetos possa ser aplicado a uma imagem de teste (em tempo de aplicação), os pontos-chave das imagens de treinamento devem ser armazenados, e seus descritores, indexados. Inicialmente, pontos-chave SIFT são extraídos de cada imagem de treinamento. Então, os descritores dos pontos-chave das imagens de treinamento são indexados com uso de uma árvore hierárquica $k$-médias, proposta por Muja e Lowe (2009).

Em tempo de aplicação, o algoritmo recebe, como entrada, uma imagem de teste. O reconhecimento de objetos ocorre, então, em um processo de três etapas:

1. **Estabelecimento de correspondências entre pontos-chave**. Inicialmente, pontos-chave são extraídos da imagem de teste. Então, correspondências entre pontos-chave individuais são estabelecidas entre os pontos-chave da imagem de teste e os pontos-chave das imagens de treinamento (como ilustrado na Figura 3.1). Correspondências entre pontos-chave são encontradas de maneira eficiente mediante o uso de uma árvore que indexa os descritores dos pontos-chave de treinamento. Neste trabalho, a árvore de indexação empregada é a árvore hierárquica $k$-médias, proposta por Muja e Lowe (2009).

2. **Estabelecimento de correspondências entre grafos-chave**. Inicialmente, triplas de pontos-chave (isto é, grafos-chave) são selecionadas a partir dos pontos-chave na imagem de teste (como ilustrado na Figura 3.2). Esses grafos-chave são então utilizados para transformar as correspondências entre pontos-chave em correspondências entre grafos-chave, o que elimina, corretamente, a grande maioria das correspondências inicialmente estabelecidas entre pontos-chave (como ilustrado na Figura 3.1). Para estabelecer uma correspondência entre uma tripla de pontos-chave $(p, q, r)$ na imagem de teste e uma tripla de pontos-chave $(p', q', r')$ em uma imagem de treinamento, o primeiro requisito é que as três correspondências individuais entre pontos-chave, $(p, p')$, $(q, q')$ e $(r, r')$, tenham sido estabelecidas na primeira etapa do método. Caso essas três correspondências entre pontos-chave tenham sido estabelecidas, o método avalia as propriedades estruturais de ambos os grafos-chave, $(p, q, r)$ e $(p', q', r')$, e, se ambos

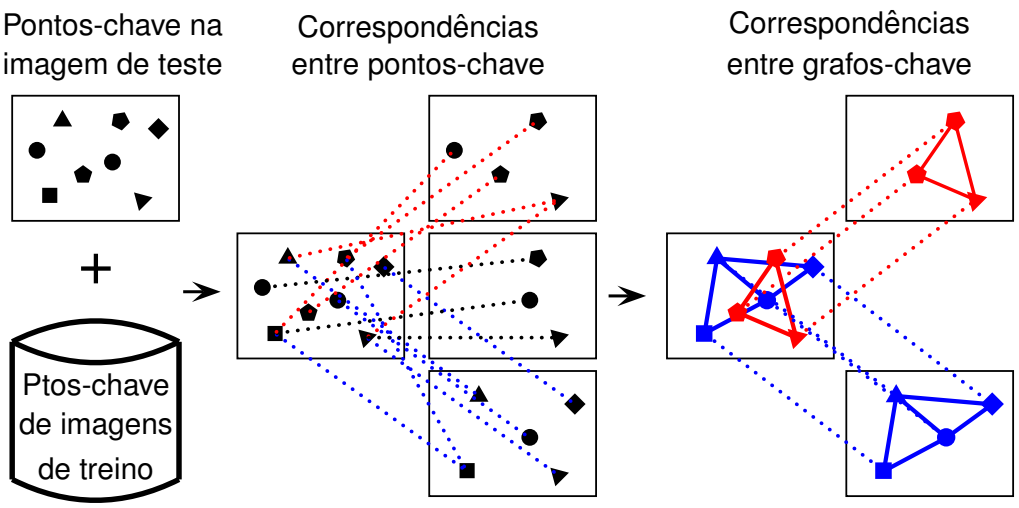


**Figura 3.1:** *Estabelecimento de correspondências entre grafos-chave na imagem de teste e na imagem de treinamento. Inicialmente, pontos-chave na imagem de teste estabelecem correspondências com pontos-chave nas imagens de treinamento. Então, as correspondências entre pontos-chave são transformadas em correspondências entre grafos-chave*

os grafos-chave não satisfizerem as propriedades estruturais, a correspondência candidata entre os grafos-chave é eliminada. Os aspectos estruturais analisados englobam o posicionamento relativo, a escala, a orientação e o sentido de ciclo dos pontos-chave (como ilustrado na Figura 3.3). Como resultado, mais de 99% das correspondências entre grafos-chave inicialmente estabelecidas (que satisfazem a co-ocorrência das triplas) são corretamente eliminadas.

3. **Estimação da pose do objeto encontrado**. Cada correspondência entre grafos-chave que tenha sido estabelecida com sucesso na etapa anterior é utilizada para gerar uma pose de objeto candidata, a qual consiste em uma uma transformação afim entre a imagem de teste e a imagem de treinamento. Cada pose candidata é avaliada de acordo com a quantidade de correspondências entre pontos-chave (vértices dos grafos-chave) que "concordam" com ela.

## 3.2 Primeiro estágio: correspondências entre pontos-chave

No primeiro estágio, muitas correspondências entre pontos-chave individuais são estabelecidas. Posteriormente, a maior parte dessas correspondências iniciais serão eliminadas mediante o uso de informações estruturais de grafos-chave.

A vantagem decorrente do emprego de uma árvore que indexa os descritores dos pontos-chave de treinamento é que cada ponto-chave $p$ da imagem de teste precisa ser comparado a somente uma pequena fração (por exemplo, $0,1\%$) do total de pontos-chave de treinamento que estão armazenados; nesse contexto, a comparação entre dois pontos-chave consiste no cálculo da distância euclidiana entre seus descritores.

### 3.2.1 Armazenamento de pontos-chave de imagens de treinamento

Antes do método de reconhecimento de objetos ser utilizado em tempo de aplicação, os pontos-chave são extraídos das imagens de treinamento e armazenados, e seus descritores locais são indexa-

dos. Para cada ponto-chave SIFT, o método armazena um identificador da imagem de treinamento na qual o ponto-chave foi extraído e também armazena os atributos desse ponto-chave: seu descritor (um vetor com 128 dimensões), sua escala, sua orientação e sua posição $x$, $y$.

O método de indexação de descritores empregado nesta tese é a árvore hierárquica $k$-médias, proposta por Muja e Lowe (2009). A árvore é construída, a partir do conjunto de todos os descritores de pontos-chave de imagens de treinamento, mediante a separação do espaço dos descritores em $k$ regiões distintas, com uso do algoritmo de aglomeração $k$-médias; então, o algoritmo $k$-médias é recursivamente aplicado aos descritores presentes em cada uma das $k$ regiões do espaço de descritores. A recursão é encerrada quando menos que $k$ descritores estejam em uma região; então, esses (menos que $k$) descritores são armazenados em uma folha da árvore.

Em tempo de aplicação, o descritor de cada ponto-chave $p$ extraído da imagem de teste percorre a árvore, a fim de estabelecer correspondências com os descritores (pontos-chave) das imagens de treinamento. O descritor de $p$ é comparado a um descritor indexado mediante o cálculo da distância euclidiana entre os vetores SIFT.

Um parâmetro fundamental que deve ser especificado é o número $L$ de descritores indexados que são comparados a um descritor de ponto-chave de imagem de teste. A vantagem essencial proporcionada pelo uso de uma árvore de indexação é que o parâmetro $L$ pode ser especificado com um valor muito menor do que o total de descritores indexados na árvore – por exemplo, nos experimentos realizados neste trabalho, foi observado que especificar $L$ como $0,1\%$ do total de descritores armazenados já possibilitava um desempenho relativamente bom no reconhecimento de objetos.

### 3.2.2 Estabelecimento de correspondências entre pontos-chave

Cada ponto-chave $p$ da imagem de teste percorre a árvore hierárquica $k$-médias a fim de estabelecer correspondências com os pontos-chave de imagens de treinamento.

Inicialmente, o ponto-chave $p$ percorre a árvore por completo, até alcançar uma folha (na qual estão armazenados menos que $k$ descritores de pontos-chave). Em cada nível intermediário da árvore, o descritor de $p$ é comparado aos $k$ vetores representativos dos aglomerados (*clusters*) presentes naquele nível, e $p$ é associado ao aglomerado que tenha resultado na menor distância euclidiana; então, as $k-1$ distâncias aos outros $k-1$ aglomerados são inseridas em uma fila de prioridades. Quando a folha da árvore é, finalmente, atingida, o descritor de $p$ é comparado a cada um dos (menos que $k$) descritores armazenados na folha. Após a folha da árvore ser analisada, o próximo elemento é retirado da fila de prioridades, o qual consiste no aglomerado (região do espaço de descritores) que apresenta a menor distância euclidiana ao descritor de $p$. Então, a árvore é novamente percorrida a partir desse aglomerado, e a travessia da árvore prossegue, recursivamente. Cada vez que um nível intermediário da árvore é analisado (isto é, uma região do espaço de descritores dividida em $k$ aglomerados), os $k-1$ aglomerados que não foram explorados são inseridos na fila de prioridades. O algoritmo encerra quando um total de $L$ descritores de pontos-chave de treinamento (que estão armazenados nas folhas das árvores) são comparados ao descritor de $p$.

O método originalmente proposto por Muja e Lowe (2009) para utilizar a árvore hierárquica $k$-médias tinha como objetivo encontrar somente *um* vizinho mais próximo ao ponto-chave $p$; isto é, tinha como meta encontrar o ponto-chave $p'$ (de uma imagem de treinamento) cujo descritor é o vizinho mais próximo ao descritor de $p$ no espaço dos descritores. Já nesta tese, uma estratégia

diferente é empregada: o objetivo é encontrar o vizinho mais próximos ao descritor de $p$ em diferentes imagens de treinamento. Assim, o ponto-chave $p$ estabelece uma correspondência com, no máximo, *um* ponto-chave $p'$ em *cada* imagem de treinamento $I'$, em que $p'$ é o ponto-chave na imagem $I'$ cujo descritor apresentou a menor distância euclidiana ao descritor de $p$, dentre todos os pontos-chave da imagem $I'$ que foram eventualmente comparados ao ponto-chave $p$. Experimentalmente, foi observado que essa limitação de um máximo de uma correspondência entre $p$ e cada imagem de treinamento acarretava na perda de poucas correspondências corretas, enquanto evitava que muitas poses de objeto candidatas incorretas fossem avaliadas na etapa final de estimação de pose.

## 3.3 Segundo estágio: correspondências entre grafos-chave

No segundo estágio do processo de reconhecimento de objetos, inicialmente, grafos-chave (triplas de pontos-chave) são selecionados na imagem de teste. Então, esses grafos-chave estabelecem correspondências candidatas com possíveis grafos-chave nas imagens de treinamento. Um grande número de correspondências entre grafos-chave candidatas são encontradas; em seguida, cada uma dessas correspondência entre grafos-chave candidatas precisa satisfazer propriedades estruturais, a fim de ser considerada válida.

Um grafo-chave é definido como um grafo $G = (V, E)$, em que o conjunto de vértices $V$ é composto de pontos-chave extraídos em uma mesma imagem e o conjunto $E$ é composto de arestas (isto é, pares ordenados de vértices). Cada grafo-chave $G$ possui $\kappa$ vértices e consiste em um *circuito orientado no sentido anti-horário*, $G = (v_1, v_2, \ldots, v_\kappa)$, como definido originalmente por Morimitsu *et al.* (2011). Cada aresta é representada, em uma imagem, como uma linha reta conectando um par de pontos-chave. Assim, as arestas de um grafo-chave $G = (v_1, v_2, \ldots, v_\kappa)$ conectam $v_1$ a $v_2$, $v_2$ a $v_3$, ..., $v_\kappa$ a $v_1$, no sentido anti-horário.

Neste trabalho, são empregados grafos-chave compostos de $\kappa = 3$ vértices (pontos-chave). Isso possibilita que *uma* correspondência entre grafos-chave acarrete em $\kappa = 3$ correspondências entre pontos-chave individuais, o que é suficiente para instanciar uma transformação afim mapeando a imagem de teste e a imagem de treinamento.

### 3.3.1 Seleção de grafos-chave a partir de pontos-chave na imagem de teste

A obtenção de grafos-chave a partir de pontos-chave na imagem de teste envolve selecionar um subconjunto do total de triplas de pontos-chave possíveis.

O algoritmo RANSAC de avaliação de poses sorteia, de forma aleatória, uma correspondência entre pontos-chave por vez, até que três correspondências sejam selecionadas; então, essas três correspondências entre pontos-chave são usadas para instanciar uma pose de objeto candidata (uma transformação afim). Dessa forma, iterativamente, o algoritmo RANSAC avalia uma pose de objeto candidata por vez, até que uma quantidade máxima de poses seja avaliada. Assim, a quantidade de poses de objeto que são avaliadas pelo algoritmo RANSAC cresce *cubicamente* com o número total de correspondências entre pontos-chave. Já nesta tese, uma estratégia mais elaborada de seleção de triplas de pontos-chave é proposta, a qual emprega triangulações de Delaunay complementares; isso permite que a quantidade de poses de objeto que são avaliadas cresça *linearmente* com o total de correspondências entre pontos-chave.

A estratégia de seleção de triplas de pontos-chave na imagem de teste proposta nesta tese é

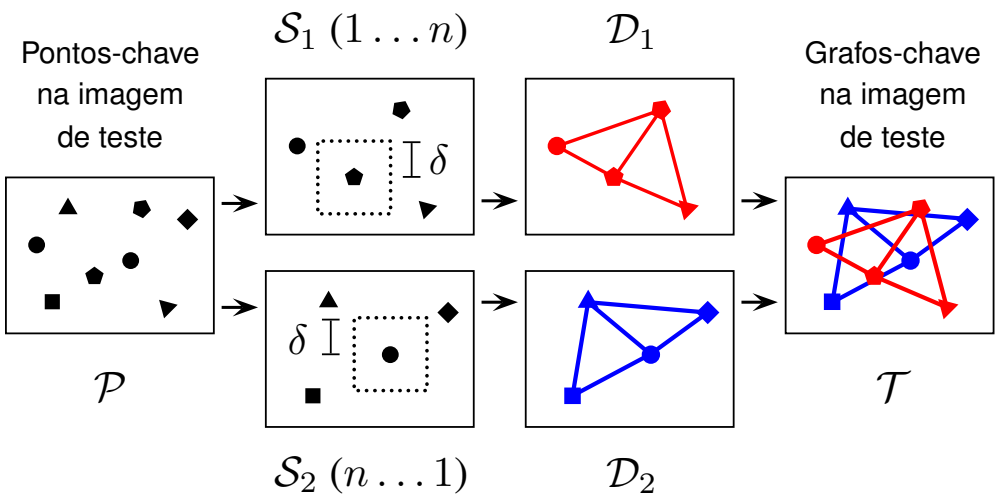


**Figura 3.2:** *Seleção de triplas de pontos-chave (grafos-chave) na imagem de teste. Inicialmente, um conjunto de pontos-chave $\mathcal{P}$ é detectado na imagem de teste. Então, subconjuntos complementares de pontos chave $\mathcal{S}_1$ e $\mathcal{S}_2$ são selecionados (com uso de uma distância $\ell_\infty$ mínima entre pontos-chave de $\delta$* pixels*), sendo empregados para calcular triangulações de Delaunay $\mathcal{D}_1$ e $\mathcal{D}_2$. Por fim, as triplas de pontos-chave $\mathcal{T} = \mathcal{D}_1 \cap \mathcal{D}_2$ são retornadas.*

realizada mediante um processo de duas etapas. Na primeira etapa, a partir do conjunto completo de pontos-chave na imagem de teste, $\mathcal{P}$, um subconjunto $\mathcal{S}_i$ de pontos-chave é selecionado, de forma que, nesse subconjunto $\mathcal{S}_i$ selecionado, não existam dois pontos-chave que estejam demasiadamente próximos entre si, na imagem. Na segunda etapa, uma triangulação de Delaunay $\mathcal{D}_i$ é calculada a partir dos pontos-chave anteriormente selecionados $\mathcal{S}_i$, o que produz, efetivamente, triplas de pontos-chave. Esse processo de geração de triplas é repetido $T$ vezes, e todas as triplas de pontos-chave geradas (em todas as $T$ execuções) são utilizadas conjuntamente; isto é, o conjunto final de triplas é dado por $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \ldots \cup \mathcal{D}_T$.

Um subconjunto de pontos-chave $\mathcal{S}_i \subseteq \mathcal{P}$ é selecionado de forma que os pontos-chave não estejam muito próximos entre si, na imagem, o que evita que os grafos-chave gerados pela triangulação de Delaunay tenham arestas muito curtas (o que aumentaria a sensibilidade a ruído). Como a quantidade de pontos-chave em um único subconjunto, $|\mathcal{S}_i|$, é possivelmente muito menor que a quantidade total de pontos-chave na imagem de teste, $|\mathcal{P}|$, um total de $T$ subconjuntos diferentes é utilizado, $\mathcal{S}_1, \ldots, \mathcal{S}_T$, virtualmente garantindo que cada pontos-chave em $\mathcal{P}$ seja selecionado ao menos uma vez.

### Seleção do primeiro subconjunto de pontos-chave

A seleção do primeiro subconjunto de pontos-chave $\mathcal{S}_1 \subseteq \mathcal{P}$ ocorre da seguinte forma. Inicialmente, $\mathcal{S}_1$ é um conjunto vazio. Então, pontos-chave $p_j$ são aleatoriamente acessados em $\mathcal{P}$, e uma tentativa é feita para incluir $p_j$ no conjunto $\mathcal{S}_1$ atualmente sob construção. A inclusão de pontos-chave em $\mathcal{S}_1$ encerra quando todos os pontos-chave em $\mathcal{P}$ tenham sido acessados. Para que um ponto-chave $p_j$ seja incluído no conjunto $\mathcal{S}_1$ sob construção, a distância $\ell_\infty$, em *pixels*, entre $p_j$ e cada ponto-chave anteriormente incluído em $\mathcal{S}_1$ deve ser superior a $\delta = 8$ *pixels*.

A seleção aleatória de pontos-chave que são incluídos no primeiro subconjunto $\mathcal{S}_1$ é feita mediante

o uso de um *array* (vetor implementado em uma linguagem de programação) $A$, que contém todos os pontos-chave $\mathcal{P}$ na imagem de teste. Inicialmente, os pontos-chave no *array* $A$ são embaralhados, aleatoriamente. Então, para selecionar o primeiro subconjunto de pontos-chave $\mathcal{S}_1$, a sequência de pontos-chave que são acessados em $A$ é dada iniciando-se a partir da *primeira* posição de $A$ (isto é, a partir do ponto-chave armazenado na primeira posição de $A$), e então avançando uma posição em $A$ a cada vez, até que a última posição em $A$ seja acessada. Cada vez que a $j$-ésima posição no *array* $A$ é acessada, o método faz uma tentativa de incluir, no subconjunto $\mathcal{S}_1$ sendo construído, o ponto-chave $p_j$ nessa $j$-ésima posição do *array* $A$.

A complexidade para selecionar o subconjunto de pontos-chave $\mathcal{S}_1$ é $O(n)$, em que $n$ é a quantidade de pontos-chave na imagem de teste. Embaralhar os pontos-chave no *array* $A$ tem complexidade $O(n)$. Então, os pontos-chave em $A$ são incluídos em $\mathcal{S}_1$. A tentativa de incluir cada ponto-chave de $A$ em $\mathcal{S}_1$ tem complexidade $O(1)$, graças ao uso de uma grade com células quadradas, em que o comprimento do lado de cada uma dessas células quadradas é $\delta = 8$ *pixels*; tentar incluir um ponto-chave em uma célula da grade envolve checar se essa célula já está ocupada por um ponto-chave, e, caso não esteja, deve ser checada a distância $\ell_\infty$ entre o ponto-chave que se deseja incluir e um possível ponto-chave em cada uma das oito células vizinhas.

### Seleção dos próximos subconjunto de pontos-chave

Após a criação do primeiro subconjunto de pontos-chave $\mathcal{S}_1 \subseteq \mathcal{P}$, o segundo subconjunto $\mathcal{S}_2 \subseteq \mathcal{P}$ é construído. O mesmo *array* $A$ de pontos-chave que foi utilizado na seleção de $\mathcal{S}_1$ (isto é, com o mesmo embaralhamento de posições) é também empregado na seleção de $\mathcal{S}_2$, mas, para selecionar $\mathcal{S}_2$, as posições de $A$ são acessadas a partir da *última* posição e então decrescendo uma posição por vez, até chegar à primeira posição de $A$.

A fim de obter $T$ subconjunto de pontos-chave, $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_T$, o procedimento descrito para obter $\mathcal{S}_1$ e $\mathcal{S}_2$ é, simplesmente, repetido. Cada par de subconjuntos $\mathcal{S}_i$ e $\mathcal{S}_{i+1}$ emprega o mesmo *array* $A_i$ de pontos-chave embaralhados. Por exemplo, $\mathcal{S}_1$ e $\mathcal{S}_2$ utilizam $A_1$, enquanto $\mathcal{S}_3$ e $\mathcal{S}_4$ utilizam $A_3$, em que $A_1$ e $A_3$ consistem em *arrays* contendo os pontos-chave da imagens de teste, cada um (aleatoriamente) embaralhado de uma forma distinta.

A vantagem de empregar a estratégia proposta para selecionar subconjuntos de pontos-chave é explicada a seguir. Durante a construção do primeiro subconjunto $\mathcal{S}_1$, quando um ponto-chave $p_j$ é acessado (e uma tentativa é feita para incluí-lo em $\mathcal{S}_1$), os pontos-chave acessados antes de $p_j$ foram, em sequência, $p_1, p_2, \ldots, p_{j-1}$. Já durante a construção do segundo subconjunto de pontos-chave $\mathcal{S}_2$, quando o mesmo ponto-chave $p_j$ é acessado, os pontos-chave acessados antes de $p_j$ foram, em sequência, $p_n, p_{n-1}, p_{n-2}, \ldots, p_{j+1}$. Isso eleva, efetivamente, a diversidade das triplas de pontos-chave produzidas, já que os conjuntos $\{p_1, \ldots, p_{j-1}\}$ e $\{p_{j+1}, \ldots, p_n\}$ são disjuntos.

### Geração de triplas de pontos-chave com uso de triangulações de Delaunay

Após os $T$ subconjuntos de pontos-chave serem selecionados, cada subconjunto $\mathcal{S}_i$ é empregado para calcular uma triangulação de Delaunay $\mathcal{D}_i$, a partir da posição $x$, $y$ (na imagem) de cada ponto-chave em $\mathcal{S}_i$. Isso gera o conjunto de triplas de pontos-chave $\mathcal{D}_i = \{(p_1, p_2, p_3), (r_1, r_2, r_3), \ldots\}$, em que cada tripla de pontos-chave em $\mathcal{D}_i$ é utilizada como um grafo-chave. O cálculo de uma triangulação de Delaunay $\mathcal{D}_i$ possui complexidade $O(n \log n)$ e produz $O(n)$ triplas de pontos-chave.

O conjunto de todas as triplas de pontos-chave selecionadas na imagem de teste é dado por $\mathcal{T} = \mathcal{D}_1 \cup \ldots \cup \mathcal{D}_T$. Cada tripla de pontos-chave em $\mathcal{T}$ é inserida em uma tabela *hash* a fim de checar se essa tripla já foi selecionada anterioremente, o que evita que triplas repetidas sejam empregadas; tal checagem possui complexidade $O(1)$.

Em resumo, o cálculo do conjunto $\mathcal{T}$ de grafos-chave em uma imagem de teste possui complexidade $O(Tn \log n)$ e gera um número de grafos-chave (triplas de pontos-chave) que é igual a $|\mathcal{T}| = O(Tn)$.

### 3.3.2 Estabelecimento de correspondências entre grafos-chave

Os grafos-chave nas imagens de treinamento não são gerados como os grafos-chave na imagem de teste, mediante o uso de triangulações de Delaunay. Para gerar grafos-chave na imagem de treinamento, primeiro, grafos-chave em potencial são obtidos mediante o uso dos grafos-chave encontrados na imagem de teste e das correspondências entre pontos-chave que foram estabelecidas no primeiro estágio do processo de reconhecimento de objetos. Isso gera correspondências entre grafos-chave candidatas; em seguida, as propriedades estruturais de grafos-chave são empregadas a fim de eliminar algumas dessas correspondências candidatas, que apresentem uma alta probabilidade de serem falsas.

#### Estabelecimento inicial de correspondências entre grafos-chave candidatas

Um grafo-chave na imagem de teste pode estabelecer no máximo *uma* correspondência com cada imagem de treinamento. Seja $G = (v_1, v_2, v_3)$ um grafo-chave na imagem de teste; $G$ pode estabelecer uma correspondência com cada imagem de treinamento com a qual seus três vértices (pontos-chave) tenham estabelecido correspondência. Assuma que, durante o estágio que estabelece correspondências entre pontos-chave individuais, as três correspondências entre pontos-chave foram encontradas, $(v_1, v_1')$, $(v_2, v_2')$ e $(v_3, v_3')$, entre o grafo-chave $G$ (na imagem de teste $I$) e pontos-chave em uma imagem de treinamento $I'$. Dessa forma, a correspondência entre grafos-chave candidata é $(G, H)$, em que $H = (u_1, u_2, u_3)$ é um grafo-chave candidato na imagem de treinamento $I'$. Assim, essa correspondência entre grafos-chave candidata $(G, H)$ está associada às correspondências entre pontos-chave individuais $\mathcal{M} = \{(v_1, u_1), (v_2, u_2), (v_3, u_3)\}$.

A execução desse estágio inicial de estabelecimento de correspondências candidatas entre grafos-chave envolve checar, para cada grafo-chave $G = (v_1, v_2, v_3)$ na imagem de teste, com quais imagens de treinamento todos os três pontos-chave, $v_1$, $v_2$ e $v_3$, estabeleceram, inicialmente, correspondências entre pontos-chave. Para realizar essa verificação, são utilizadas, para cada ponto-chave $p$ na imagem de teste, palavras binárias com 64 *bits*, que indicam com quais imagens de treinamento $p$ estabeleceu, inicialmente, uma correspondência entre pontos-chave. Assim, as imagens de treinamento que estabeleceram uma correspondência inicial com o grafo-chave $G = (v_1, v_2, v_3)$ são obtidas com uso de uma operação binária AND entre as palavras binárias associadas a $v_1$, $v_2$ e $v_3$; os *bits* de valor 1 presentes na palavra binária resultante da operação AND podem ser acessados, de forma eficiente, mediante uma operação binária que retorna a posição, na palavra binária, de cada *bit* de valor 1 (operação denominada *count leading zeros*).

Finalmente, se a correspondência entre grafos-chave $(G, H)$ satisfizer as propriedades estruturais, ela é considerava uma correspondência válida; nesse caso, $(G, H)$ produz, como resultado, as correspondências entre pontos-chave $\mathcal{M} = \{(v_1, u_1), (v_2, u_2), (v_3, u_3)\}$.

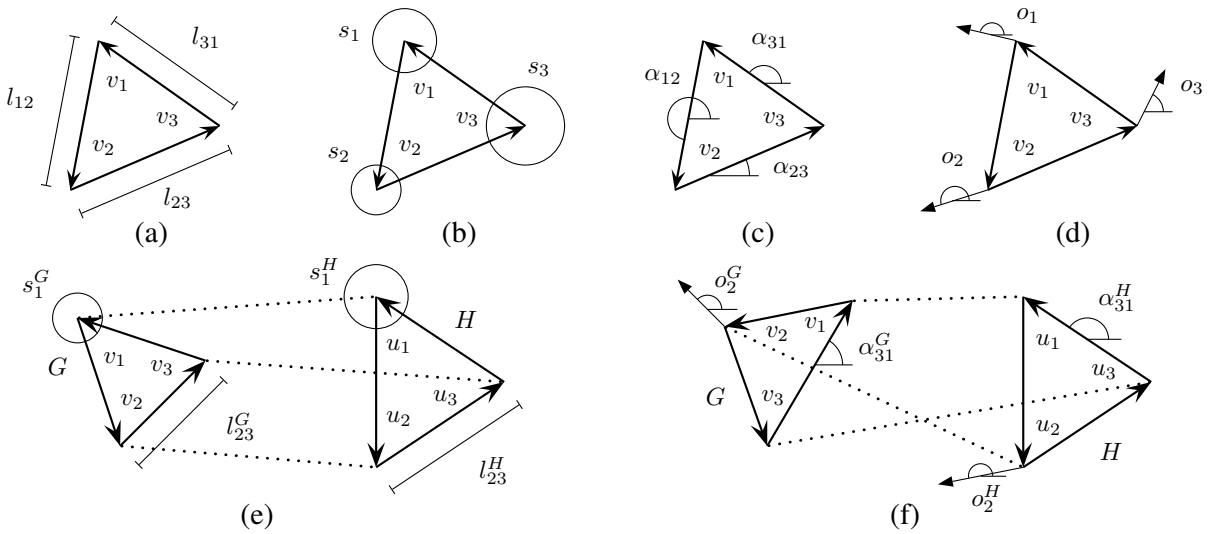


**Figura 3.3:** *Informações estruturais usadas para eliminar correspondências entre grafos-chave que provavelmente são falsas.* ***(a)*** *Comprimento das arestas dos grafos-chave: $l_{12}$, $l_{23}$ e $l_{31}$.* ***(b)*** *Escala dos pontos-chave (SIFT): $s_1$, $s_2$ e $s_3$.* ***(c)*** *Orientação das arestas dos grafos-chave: $\alpha_{12}$, $\alpha_{23}$ e $\alpha_{31}$.* ***(d)*** *Orientação dos pontos-chave (SIFT): $o_1$, $o_2$ e $o_3$.* ***(e)*** *Variações consistentes na escala dos pontos-chave e no comprimento das arestas dos grafos-chave. Para as escalas correspondentes $s_1^G$ (no grafo-chave $G$ na imagem de teste) e $s_1^H$ (no grafo-chave $H$ na imagem de treinamento), a razão $\Phi_1 = s_1^G/s_1^H$ é calculada, assim como a razão entre os comprimentos das arestas correspondentes $\Phi_{23} = l_{23}^G/l_{23}^H$. O método verifica se $\Phi_1$ é similar a $\Phi_{23}$: $0.5 \leq \Phi_1/\Phi_{23} \leq 2$. Uma verificação semelhante é feita para cada par $\Phi', \Phi''$ no conjunto de razões $\{\Phi_1, \Phi_2, \Phi_3, \Phi_{12}, \Phi_{23}, \Phi_{31}\}$.* ***(f)*** *Variações consistentes na orientação dos pontos-chave e nas arestas dos grafos-chave. Para as orientações de pontos-chave correspondentes $o_2^G$ e $o_2^H$, a variação na orientação $\Delta_2 = o_2^G - o_2^H$ é calculada, assim como a variação na orientação das arestas correspondentes $\Delta_{31} = \alpha_{31}^G - \alpha_{31}^H$. O método verifica se $\Delta_2$ é similar a $\Delta_{31}$: $\arccos(\Delta_2 - \Delta_{31}) \leq 60°$. Uma verificação semelhante é feita para cada par $\Delta', \Delta''$ no conjunto de variações na orientação $\{\Delta_1, \Delta_2, \Delta_3, \Delta_{12}, \Delta_{23}, \Delta_{31}\}$.*

### Eliminação de correspondências entre grafos-chave com uso de informações estruturais

Cada correspondência entre grafos-chave $(G, H)$ precisa satisfazer as propriedades estruturais de grafos-chave a fim de ser considerada como uma correspondência válida.

O *sentido de ciclo* é uma propriedade topológica, a qual é invariante a a uma homografia entre imagens 2D. Cada grafo-chave $G$ na imagem de teste é um circuito orientado cujo sentido de ciclo é assumido como sendo o anti-horário; já que um espelhamento não é uma transformação possível de ocorrer entre a imagem de teste e a imagem de treinamento, então o circuito de um grafo-chave $H$ na imagem de treinamento também precisa estar no sentido anti-horário. No caso de um grafo-chave candidato $H$ na imagem de treinamento estar no sentido horário, então tanto $H$ quanto a correspondência candidata $(G, H)$ são rejeitados. Para que seja possível assumir que ambos os grafos-chave, $G$ e $H$, estejam no sentido anti-horário, é necessário que o objeto de treinamento seja aproximadamente convexo ou que haja uma imagem de treinamento cujo ponto-de-vista não seja demasiado diferente do ponto-de-vista em que o objeto aparece na imagem de teste.

Outros quatro atributos são calculados para cada grafo-chave:

- **Comprimento das arestas**: Uma aresta de um grafo-chave é uma linha reta conectando um par de pontos-chave na imagem e possui um comprimento em *pixels*. Um grafo-chave $U = (h_1, h_2, h_3)$ possui três arestas, $e_{12} = (h_1, h_2)$, $e_{23} = (h_2, h_3)$ e $e_{31} = (h_3, h_1)$, cujos comprimentos são, respectivamene, $l_{12}^U$, $l_{23}^U$ e $l_{31}^U$ (como ilustrado na Figura 3.3-a).

- **Escala dos pontos-chave**: Cada ponto-chave no grafo-chave $U = (h_1, h_2, h_3)$ possui um valor de escala atribuído pelo detector de pontos-chave SIFT, cujos valores são, respectivamente, $s_1^U$, $s_2^U$ e $s_3^U$ (como ilustrado na Figura 3.3-b).

- **Comprimento das arestas**: Cada aresta de um grafo-chave possui um ângulo (orientação) em relação ao eixo horizontal da imagem; uma aresta de grafo-chave é tratada como um vetor convencional em 2D [1]. As orientações das arestas de um grafo-chave $U = (h_1, h_2, h_3)$ são, respectivamente, $\alpha_{12}^U$, $\alpha_{23}^U$ e $\alpha_{31}^U$ (como ilustrado na Figura 3.3-c).

- **Orientação dos pontos-chave**: Cada ponto-chave no grafo-chave $U = (h_1, h_2, h_3)$ possui uma orientação atribuída pelo detector de pontos-chave SIFT, as quais são, respectivamente, $o_1^U$, $o_2^U$ e $o_3^U$ (como ilustrado na Figura 3.3-d).

Para cada correspondência entre grafos-chave $(G, H)$ candidata, o método calcula as variações que ocorrem nas orientações, comprimentos e escalas, de $G$ para $H$. Uma correspondência entre grafos-chave válida deve apresentar uma variação consistente na orientação das arestas e na orientação dos pontos-chave, assim como deve apresentar uma variação consistente no comprimento das arestas e na escala dos pontos-chave.

Assuma que a correspondência candidata entre grafos-chave, $(G, H)$, esteja associada às correspondências entre pontos-chave $\mathcal{M} = \{(v_1, u_1), (v_2, u_2), (v_3, u_3)\}$, como ilustrado na Figura 3.3-e. As variações de atributos de grafos-chave são calculadas como:

- **Variações no comprimento das arestas**: Na correspondência entre grafos-chave $(G, H)$, há três razões entre os comprimentos das arestas correspondentes:

$$\Phi_{12} = \frac{l_{12}^G}{l_{12}^H}, \tag{3.1}$$

$$\Phi_{23} = \frac{l_{23}^G}{l_{23}^H}, \tag{3.2}$$

$$\Phi_{31} = \frac{l_{31}^G}{l_{31}^H}. \tag{3.3}$$

- **Variações na escala dos pontos-chave**: Na correspondência entre grafos-chave $(G, H)$, há três razões entre a escala de pontos-chave correspondentes:

$$\Phi_1 = \frac{s_1^G}{s_1^H}, \tag{3.4}$$

$$\Phi_2 = \frac{s_2^G}{s_2^H}, \tag{3.5}$$

$$\Phi_3 = \frac{s_3^G}{s_3^H}. \tag{3.6}$$

- **Variações na orientação das arestas**: Na correspondência entre grafos-chave $(G, H)$, há

[1] A origem de uma imagem é seu canto inferior-esquerdo; o eixo $y$ aponta para cima e o eixo $x$ aponta para a direita. Ângulos positivos estão no sentido anti-horário e iniciam a partir do eixo horizontal da imagem.

três diferenças entre a orientação de arestas correspondentes:

$$\Delta_{12} = \alpha_{12}^{G} - \alpha_{12}^{H}, \tag{3.7}$$

$$\Delta_{23} = \alpha_{23}^{G} - \alpha_{23}^{H}, \tag{3.8}$$

$$\Delta_{31} = \alpha_{31}^{G} - \alpha_{31}^{H}. \tag{3.9}$$

- **Variações na orientação dos pontos-chave**: Na correspondência entre grafos-chave $(G, H)$, há três diferenças entre a orientação de pontos-chave correspondentes:

$$\Delta_{1} = o_{1}^{G} - o_{1}^{H}, \tag{3.10}$$

$$\Delta_{2} = o_{2}^{G} - o_{2}^{H}, \tag{3.11}$$

$$\Delta_{3} = o_{3}^{G} - o_{3}^{H}. \tag{3.12}$$

**Similaridade nas variações de escala dos pontos-chave e de comprimento das arestas dos grafos-chave**. Quando uma transformação visual de *zoom* é aplicada a uma imagem, que muda a escala da imagem em um fator $r$, tanto o comprimento de cada aresta de grafo-chave quanto a escala de cada ponto-chave são multiplicados por $r$. Já quando uma homografia é aplicada a uma imagem, a qual altera o ângulo de visão em $\psi$ graus, um círculo unitário torna-se uma elipse, em que o eixo mais longo tem comprimento 1 e o eixo mais curto tem comprimento $\cos\psi$ (como demonstrado por Sattler *et al.* (2009)).

Características locais SIFT perdem confiabilidade quando $\psi > 60°$, como discutido em Lowe (2004). Quando $\psi = 60°$, a razão entre o maior eixo e o menor eixo da elipse transformada é $\cos 60°/1 = 0.5$.

Em uma correspondência entre grafos-chave $(G, H)$, sejam as três razões entre o comprimento de arestas correspondentes $\Phi_{12}$, $\Phi_{23}$ e $\Phi_{31}$, enquanto as três razões entre a escala de pontos-chave correspondentes são $\Phi_1$, $\Phi_2$ e $\Phi_3$. Como ilustrado na Figura 3.3-e, uma correspondência entre grafos-chave válida deve satisfazer a seguinte propriedade: cada cada par $\Phi', \Phi''$ selecionado a partir de duas dessas seis razões $\{\Phi_{12}, \Phi_{23}, \Phi_{31}, \Phi_1, \Phi_2, \Phi_3\}$, a razão de maior valor deve ser menor que o dobro da razão de menor valor, isto é,

$$0.5 \leq \frac{\Phi'}{\Phi''} \leq 2. \tag{3.13}$$

**Similaridade nas variações de orientação dos pontos-chave e de orientação das arestas dos grafos-chave**. Quando uma imagem é rotacionada em $\theta$ graus, tanto a orientação de cada ponto-chave quanto a orientação de cada aresta de grafo-chave também são rotacionadas nos mesmos $\theta$ graus. Já quando uma homografia é aplicada a uma imagem, nem todas as arestas e pontos-chave têm sua orientação rotacionada em $\theta$ graus, embora variações de orientação muito diferentes de $\theta$ graus não ocorram.

Em uma correspondência entre grafos-chave $(G, H)$, sejam as variações na orientação das arestas dos grafos-chave $\Delta_{12}$, $\Delta_{23}$ e $\Delta_{31}$, enquanto as variações na orientação dos pontos-chave são $\Delta_1$, $\Delta_2$ e $\Delta_3$. Como ilustrado na Figura 3.3-f, uma correspondência entre grafos-chave válida deve satisfazer

a seguinte propriedade: cada cada par $\Delta', \Delta''$ selecionado a partir de duas dessas seis variações de orientação $\{\Delta_{12}, \Delta_{23}, \Delta_{31}, \Delta_1, \Delta_2, \Delta_3\}$, o ângulo entre $\Delta'$ e $\Delta''$ deve ser menor que $60°$, isto é,

$$\arccos(\Delta' - \Delta'') \leq 60°. \tag{3.14}$$

## 3.4 Terceiro estágio: estimação da pose do objeto

No terceiro estágio do processo de reconhecimento de objetos, as correspondências entre grafos-chave que foram estabelecidas no estágio anterior são empregadas para localizar os objetos e estimar sua pose.

Uma correspondência entre grafos-chave gera $\kappa = 3$ correspondências entre pontos-chave, o que é suficiente para instanciar uma pose de objeto (transformação afim) candidata mapeando as imagens de teste e de treinamento. Cada pose de objeto candidata é avaliada mediante a contagem do número de correspondências entre pontos-chave (isto é, correspondências entre vértices de grafos-chave) que "concordam" acerca da instanciação da imagem de treinamento na imagem de teste.

### 3.4.1 Transformações afim candidatas

Seja $\mathcal{G}$ o conjunto de correspondências entre grafos-chave que ocorrem entre a imagem de teste e uma imagem de treinamento, $\mathcal{G} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{|\mathcal{G}|}\}$, em que $\mathcal{M}_i$ é um conjunto de três correspondências entre pontos-chave (que estão associadas a uma correspondência entre grafos-chave). Então, o conjunto de correspondências entre *pontos-chave* que ocorrem entre essas duas imagens é $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \ldots \cup \mathcal{M}_{|\mathcal{G}|}$.

A fim de avaliar o nível de confiança associado a uma transformação afim candidata, é utilizado o número de correspondências entre pontos-chave (vértices de grafos-chave). Para cada correspondência entre pontos-chave $(p, q) \in \mathcal{M}$ (em que $p$ é um ponto-chave na imagem de teste e $q$ é um ponto-chave na imagem de treinamento), seja $x_p, y_p$ a posição na imagem de teste determinada pela correspondência $(p, q)$, e seja $f(x_q), f(y_q)$ a posição na imagem de teste prevista pela transformação afim sob avaliação. Se a distância $\ell_\infty$ entre a posição $x_p, y_p$ e a posição $f(x_q), f(y_q)$ estiver abaixo de *três pixels*, é assumido que a correspondência entre pontos-chave $(p, q)$ "concorda" com a transformação afim sob avaliação. Então, o nível de confiança associado a uma transformação afim é definido como sendo a quantidade de correspondências entre pontos-chave que "concordam" com essa transformação afim.

Cada transformação afim que apresente ao menos uma correspondência entre pontos-chave que "concorde" com a instanciação é considerada uma transformação afim candidata. Cada transformação afim candidata é, então, refinada, com uso do método dos mínimos quadrados (*least squares fitting*).

### 3.4.2 Ocorrência de múltiplas detecções de objetos

O estágio final do processo de reconhecimento de objetos envolve projetar, na imagem de teste, os objetos presentes nas imagens de treinamento. Nesse contexto, é preciso lidar com a possível ocorrência de múltiplas detecção de objeto na imagem de teste.

Inicialmente, as transformações afim candidatas são ordenadas de acordo com a quantidade de correspondências entre pontos-chave que "concordam" com a instanciação (isto é, as poses de objeto candidatas são ordenadas de acordo com o nível de confiança de cada transformação afim). Então, a transformação afim que tenha o maior nível de confiança é retornada como uma detecção de objeto bem sucedida. A segmentação do objeto na imagem de treinamento encontrada é, então, mapeada para a imagem de teste, com uso da transformação afim retornada.

Em seguida, o método trata da segunda transformação afim com o maior nível de confiança. É realizada uma recontagem da quantidade de correspondências entre pontos-chave que "concordam" com essa transformação afim com o segundo maior nível de confiança. Nessa recontagem, são eliminadas correspondências entre pontos-chave que estejam, na imagem de teste, dentro da região instanciada a partir da imagem de treinamento com o maior nível de confiança (a qual já foi retornada como uma detecção bem-sucedida). No caso de ainda restar ao menos uma correspondência entre pontos-chave que "concorde" com essa segunda transformação afim sob avaliação, o método considera que ocorreu um reconhecimento de objetos bem-sucedido, e então retorna essa segunda transformação afim como uma pose de objeto encontrado. A segmentação do objeto na imagem de treinamento encontrada é, então, mapeada para a imagem de teste, com uso da transformação afim retornada.

Enquanto ainda restarem outras transformações afim candidatas, o método faz uma tentativa de retornar cada uma delas. O método considera, uma a uma, cada uma das transformações afim restantes, de acordo com a ordenação baseada no nível de confiança de cada uma delas. Para cada transformação afim sob consideração, o método elimina as correspondências entre pontos-chave que inicialmente "concordavam" com essa transformação afim mas que, na imagem de teste, estejam dentro de uma região instanciada a partir da imagem de treinamento de qualquer uma das transformações afim retornadas anteriormente. Uma transformação afim é retornada no caso de existir ao menos uma correspondência entre pontos-chave que "concorde" com essa transformação afim (isto é, no caso de existir ao menos uma correspondência entre pontos-chave que não tenha sido eliminada devido ao fato de estar, na imagem de teste, dentro de uma região mapeada a partir da imagem de treinamento de qualquer uma das transformações afim retornadas anteriormente).

# Capítulo 4

# Resultados experimentais

## 4.1 Bases de dados utilizadas

A fim de realizar uma comparação experimental entre o método de reconhecimento de objetos apresentado nesta tese e outros métodos de reconhecimento de objetos que estejam no estado-da-arte, foi empregada uma base de dados disponibilizada para uso público, denominada CMU10, introduzida por Hsiao *et al.* (2010). Essa base de dados contém dez tipos de objetos diferentes. Ela é composta de um total de 250 imagens de treinamento (com 25 imagens de treinamento para cada tipo de objeto) e 500 imagens de teste (com 50 imagens de teste para cada tipo de objeto). Cada imagem de teste apresenta um ou dois objetos presentes, a serem detectados. A base de dados CMU10 simula, de uma forma realista, a tarefa de reconhecimento de objetos aplicada a situações no "mundo real", em que ocorrem grandes variações na iluminação da cena e no ângulo de visão dos objetos, assim como é possível que os objetos apareçam somente parcialmente visíveis. Para cada imagem, de teste ou de treinamento, está disponível a segmentação do objeto na imagem.

Com intuito de avaliar a escalabilidade do método de reconhecimento de objetos proposto, isto é, avaliar o desempenho e a eficiência do método em um cenário em que o número adicionais foram obtidas a partir da base de dados PASCAL VOC 2007 (descrita em Everingham *et al.* (2010)). Um total de 4000 imagens de treinamento foi obtido (de forma aleatória) a partir das imagens disponíveis

**Figura 4.1:** *Exemplos de imagens de teste com os objetos a serem detectados (linhas superiores) e de imagens de treinamento (linhas inferiores). Figura originalmente apresentada em Hsiao et al. (2010).*

na base de dados PASCAL VOC 2007; então, essas imagens foram utilizadas em conjunto com as 250 imagens de treinamento originalmente presentes na base de dados CMU10, gerando, assim, uma base de dados de imagens denominada CMU10+PASCAL, composta de 4250 imagens de treinamento. No caso do método de reconhecimento de objetos encontrar, em uma imagem de teste, um "objeto" localizado em alguma imagem da base de dados PASCAL, tal detecção constitui um erro, isto é, um falso positivo (não ocorre o caso em que um objeto presente na base CMU10 também esteja presente na base PASCAL).

A biblioteca VLFeat (apresentada em Vedaldi e Fulkerson (2008)) foi empregada para extrair os pontos-chave SIFT a partir de cada imagem. As imagens na base de dados CMU10 produziram um total de $2.5 \times 10^5$ pontos-chave de treinamento, enquanto as imagens na base de dados CMU10+PASCAL geraram um total de $7.5 \times 10^6$ pontos-chave de treinamento; para as imagens de treinamento na base de dados CMU10, foi empregada a segmentação do objeto presente na imagem para remover os pontos-chave que não faziam parte do objeto. Já para as imagens da base de dados PASCAL, não há segmentação de objeto, de forma que uma imagem completa atua como um "objeto de treinamento".

Para cada uma das duas bases de imagens de treinamento utilizadas, CMU10 e CMU10+PASCAL, uma árvore hierárquica $k$-means foi calculada, a fim de indexar os pontos-chave de treinamento extraídos do conjunto de imagens; ambas as árvores utilizaram o parâmetro $k$ (empregado pelo algoritmo $k$-médias) como $k = 16$.

Nos experimentos realizados, o desempenho e a eficiência alcançados com uso da base de imagens CMU10 foram comparados com o desempenho e a eficiência alcançados com uso da base de imagens CMU10+PASCAL. A base de imagens CMU10+PASCAL possui 30 vezes mais pontos-chave em comparação à base de imagens CMU10, enquanto a quantidade total de imagens é 17 vezes maior.

## 4.2 Protocolo de avaliação

Para cada objeto detectado que tenha sido retornado pelo método de reconhecimento de objetos, a transformação afim retornada é empregada para instanciar, na imagem de teste, a segmentação do objeto presente na imagem de treinamento[1]. Essa instanciação, na imagem de teste, do objeto presente na imagem de treinamento origina a região $R$, na imagem de teste. Então, é calculado o coeficiente de sobreposição que ocorre entre a região $R$ e a região $R_{\mathtt{gt}}$, a qual é a segmentação correta do objeto presente na imagem de teste (*ground truth*). Para que uma detecção de objeto seja considerada correta (isto é, um verdadeiro positivo), o coeficiente de sobreposição deve ser:

$$\frac{(R \cap R_{\mathtt{gt}})}{(R \cup R_{\mathtt{gt}})} > 0.4. \tag{4.1}$$

Duas modificações foram feitas no protocolo de avaliação originalmente empregado por Hsiao *et al.* (2010), que também avaliou seu método com uso da base de dados CMU10. A primeira modificação refere-se às segmentações, nas imagens de teste, que indicam os objeto a serem detectados. En-

[1] No caso de uma imagem de treinamento da base de imagens PASCAL ser encontrada, não há segmentação de objeto na imagem de treinamento que possa ser mapeada para a imagem de teste. Assim, na imagem de teste, a região do "objeto detectado" é considerada como sendo somente o triângulo associado ao grafo-chave na imagem de teste.

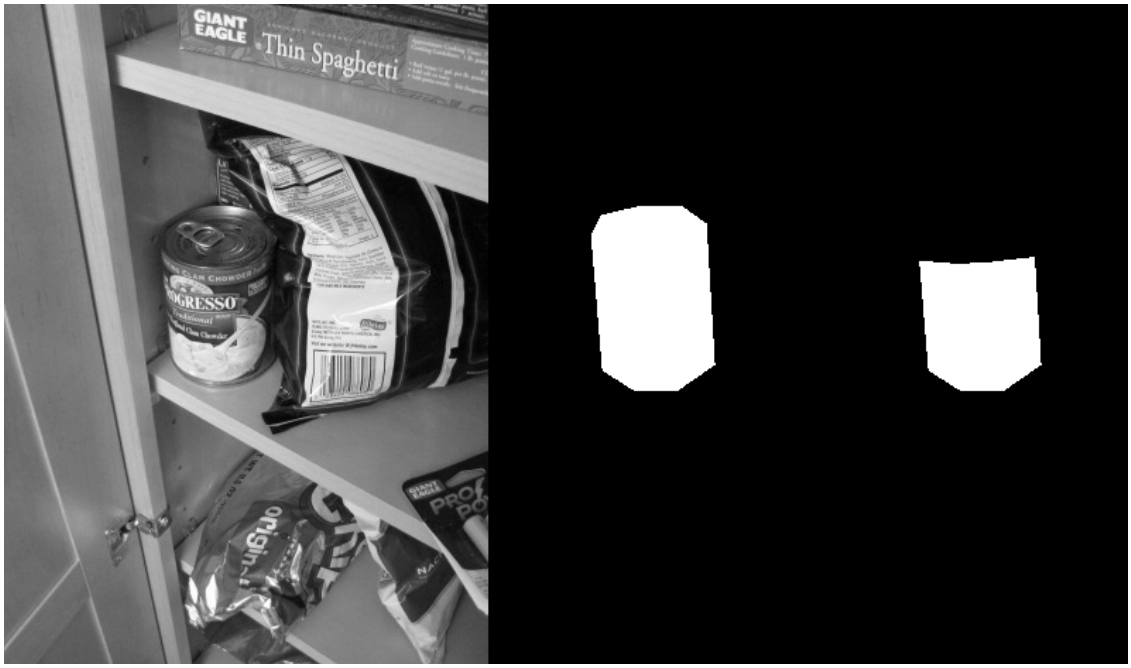


**Figura 4.2:** *Parte de uma imagem de teste (à esquerda), segmentação original do objeto a ser detectado (no centro) e segmentação modificada (à direita), em que a parte superior do objeto foi removida.*

quanto o método de Hsiao *et al.* (2010) instancia modelos tridimensionais de objetos nas imagens de teste, o método dos grafos-chave proposto neste trabalho instancia imagens de treinamento nas imagens de teste. Entretanto, nas imagens de treinamento, os objetos somente aparecem em um ponto-de-vista frontal, isto é, nem a parte superior dos objetos (como a tampa) nem a parte inferior estão, de fato, visíveis nas imagens de treinamento. Isso não constitui um problema para o método de Hsiao *et al.* (2010) devido ao uso de modelos tridimensionais, pois, quando um modelo tridimensional é instanciado na imagem de teste, essa instanciação considera essas partes superiores e inferiores dos objetos. Contudo, o uso de uma transformação afim para instanciar imagens de treinamento, como feito neste trabalho, não é capaz de considerar essas partes superiores e inferiores de objetos. Assim, a fim de realizar uma avaliação mais justa do método dos grafos-chave apresentado nesta tese, a segmentação dos objetos na imagem de teste foi ligeiramente modificada, de forma a remover partes que não eram visíveis em nenhuma imagem de treinamento (isto é, partes superiores e inferiores dos objetos). A Figura 4.2 apresenta um exemplo de (parte de) uma imagem de teste e a segmentação originalmente disponível por Hsiao *et al.* (2010) e a segmentação modificada, em que a região correspondente à tampa do objeto foi removida.

A segunda modificação feita no protocolo de avaliação originalmente empregado por Hsiao *et al.* (2010) refere-se ao limiar do coeficiente de sobreposição. Enquanto Hsiao *et al.* (2010) utilizou um limiar de 0.5 no coeficiente de sobreposição, para que uma detecção de objeto seja considerada correta, nesta tese, um limiar de 0.4 no coeficiente de sobreposição foi utilizado (como mostrado na Equação 4.2). Essa modificação foi necessária devido ao fato de que algumas detecções de objeto *corretas* atingiram um coeficiente de sobreposição entre 0.4 e 0.5, em situações nas quais o ponto-de-vista do objeto na imagem de teste era demasiado diferente do ponto-de-vista do objeto na imagem de treinamento encontrada; em comparação, o método de Hsiao *et al.* (2010) pode empregar um limiar de 0.5, pois o uso de modelos tridimensionais já engloba em uma única entidade (o modelo

3D) o ponto-de-vista de várias imagens de treinamento diferentes.

Foi observado, experimentalmente, que as duas modificações no protocolo de avaliação não acarretaram que alguma detecção incorreta fosse erroneamente contada como sendo correta.

A fim de estimar o desempenho alcançado por um método de reconhecimento de objetos, é calculada a Precisão Média (PM) para cada um dos dez tipos de objetos; então, a média dos dez valores de Precisão Média é calculada, a qual sumariza o desempenho do método de reconhecimento de objetos considerado. Para calcular um valor de Precisão Média para um tipo de objeto, um gráfico de precisão-revocação (*precision-recall*), para esse objeto considerado, é traçado, e então a área abaixo da curva de precisão-revocação constitiu o valor de Precisão Média associado ao objeto.

## 4.3 Métodos comparados

O desempenho do método de reconhecimento de objetos proposto neste trabalho foi comparado ao desempenho alcançado pelos métodos SCRAMSAC, proposto por Sattler *et al.* (2009), e a uma versão adaptada do método apresentado no artigo que propôs os pontos-chave SIFT, Lowe (2004).

Inicialmente, a árvore hierárquica $k$-médias é usada para estabelecer correspondências entre pontos-chave individuais. No método proposto por Lowe (2004) para ser usado com os pontos-chave SIFT, cada ponto-chave $p$ na imagem de teste pode estabelecer um total de uma correspondência, com algum ponto-chave de uma imagem de treinamento. Já no caso do método SCRAMSAC, um ponto-chave $p$ na imagem de teste, após percorrer a árvore hierárquica $k$-médias, pode estabelecer até uma correspondência com um ponto-chave em cada imagem de treinamento; em seguida, somente as $N$ correspondências com a menor distância euclidiana entre os descritores dos pontos-chave são mantidas (isto é, as outras correspondências são eliminadas); por fim, cada correspondência entre pontos-chave precisa satisfazer as propriedades estruturais propostas no método SCRAMSAC, para que não seja eliminada.

No estágio final, de estimação de pose dos objetos, o método RANSAC é utilizado. Para o método SCRAMSAC e o método proposto por Lowe (2004), *três* correspondências inicialmente estabelecidas entre pontos-chave com uma mesma imagem de treinamento são aleatoriamente selecionadas, o que gera uma pose de objeto (transformação afim) candidata. Já no método dos grafos-chave proposto nesta tese, *uma* correspondência entre grafos-chave é aleatoriamente selecionada a fim de gerar uma pose de objeto candidata (como discutido na Seção 3.4.1).

## 4.4 Resultados

### 4.4.1 Exemplos de detecções de objeto bem-sucedidas

A Figura 4.3 apresenta uma detecção de objetos bem sucedida, em que são apresentadas duas figuras. A figura superior mostra as correspondências entre grafos-chave que foram estabelecidas. A figura inferior mostra a transformação afim encontrada e as correspondências entre pontos-chave (vértices dos grafos-chave) que "concordam" com essa transformação afim (o coeficiente de sobreposição calculado foi 0.61). Embora todas as correspondências entre grafos-chave mostradas estejam corretas, algumas correspondências entre os vértices desses grafos-chave (isto é, correspondências entre pontos-chave) não "concordaram" com a transformação afim encontrada. Isso ocorre devido à limitação da abordagem deste trabalho para instanciar poses de objetos, que é utilizar

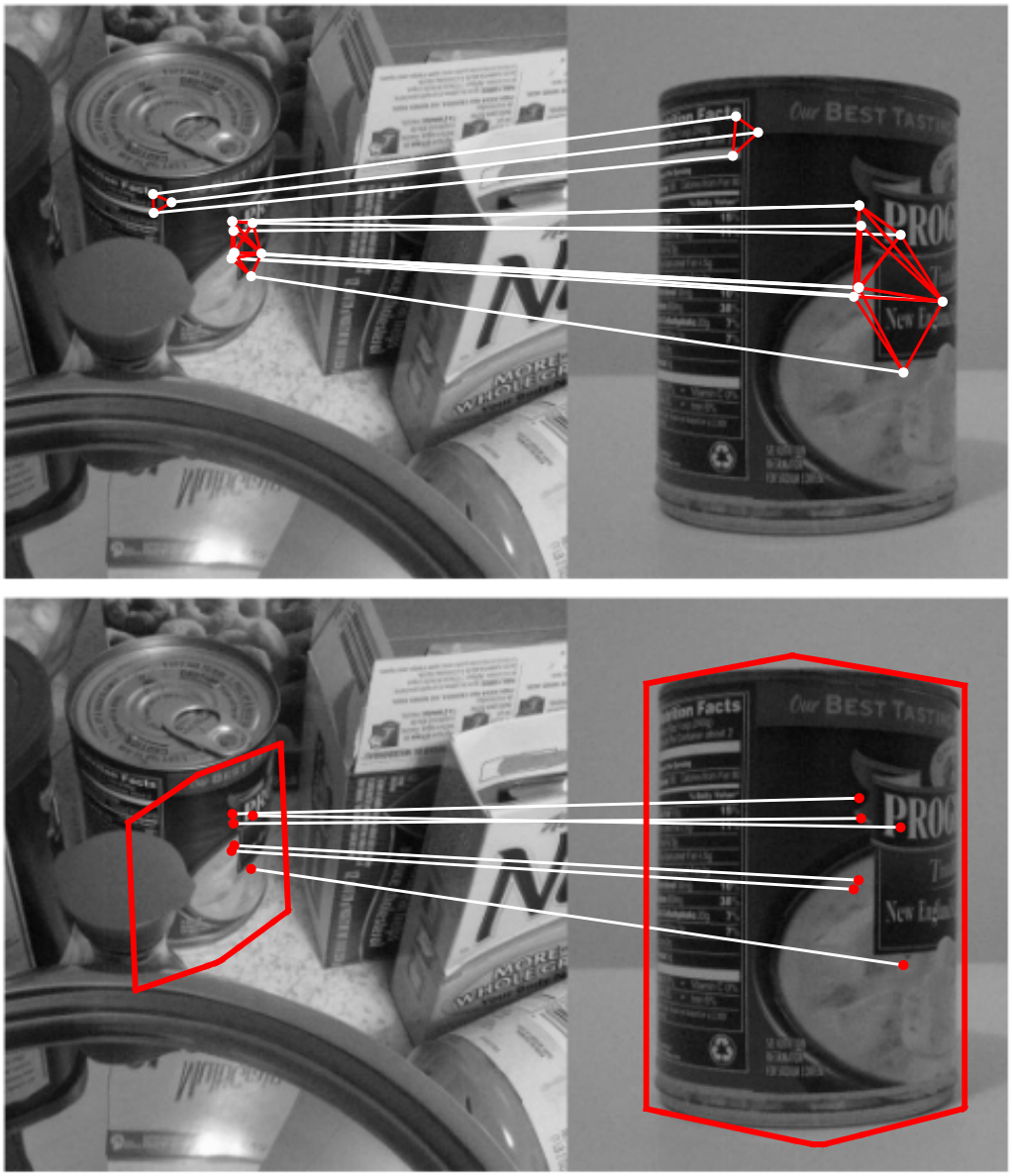

**Figura 4.3:** *Correspondências entre grafos-chave e correspondências entre pontos-chave que "concordam" com a transformação afim encontrada. O coeficiente de sobreposição calculado foi* 0.61.


**Figura 4.4:** *Correspondências entre pontos-chave que "concordam" com a transformação afim encontrada. O coeficiente de sobreposição calculado foi* 0.47.

**Figura 4.5:** *Correspondências entre pontos-chave que "concordam" com a transformação afim encontrada. O coeficiente de sobreposição calculado foi* 0.42.

transformações afim simples entre imagens. Para evitar esse problema, poderiam ser empregadas estratégias mais elaboradas de instanciação de poses de objetos, tais como o uso de modelos tridimensionais de objetos de treinamento ou a combinação das correspondências entre pontos-chave que ocorrem entre a imagem de teste e imagens de treinamento diferentes (de forma semelhante ao método proposto por Lowe (2001)).

As Figuras 4.4, 4.5 e 4.6 apresentam, cada uma, uma detecção de objetos bem sucedida. São mostradas, também, as correspondências entre pontos-chave (vértices dos grafos-chave) que "concordam" com a transformação afim encontrada. Os coeficientes de sobreposição calculados foram, respectivamente, 0.47, 0.42 e 0.62.

### 4.4.2 Quantidade de triangulações de Delaunay utilizadas

A Figura 4.7 apresenta o desempenho (estimado como a média dos dez valores de Precisão Média, abreviado como mPM) alcançado pelo método dos grafos-chave versus a quantidade de triangulações de Delaunay $T$ usadas para selecionar triplas de pontos-chave (grafos-chave) na imagem de teste,

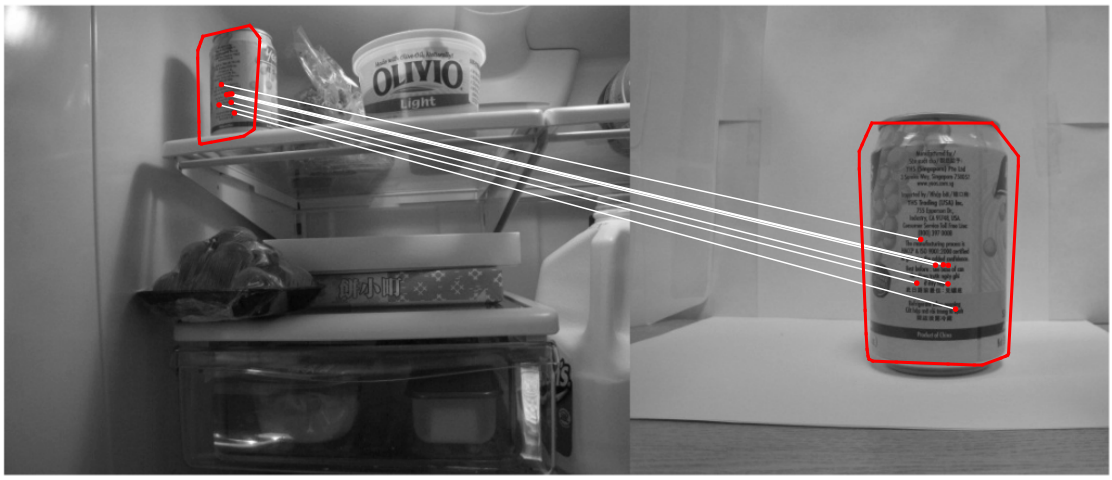


**Figura 4.6:** *Correspondências entre pontos-chave que "concordam" com a transformação afim encontrada. O coeficiente de sobreposição calculado foi* 0.62.

como discutido na seção 3.3.1. O gráfico apresenta duas curvas, uma associada ao uso da base de dados CMU10 como imagens de treinamento e a outra associada ao uso da base de dados CMU10+PASCAL como imagens de treinamento. Em ambos os casos, durante o estágio inicial de estabelecimento de correspondências entre pontos-chave com uso da árvore hierárquica $k$-médias, cada ponto-chave da imagem de teste é comparado a um total de $L = 1000$ pontos-chave de imagens de treinamento indexados (uma comparação entre pontos-chave ocorre mediante o cálculo da distância euclidiana entre os descritores, como apresentado na Seção 3.2.2).

A Figura 4.7 demonstra que o uso de muitas triangulações de Delaunay possibilitou um grande aumento no desempenho (mPM) do método dos grafos-chave: o valor de mPM aumentou em quase 0.5, em comparação ao valor de mPM alcançado com o uso de somente uma triangulação de Delaunay (isto é, $T = 1$). No caso do uso da base de dados CMU10 como imagens de treinamento, o valor de mPM alcançado não cresceu significamente com o uso de mais que $T = 50$ triangulações de Delaunay. Já quando a base de dados CMU10+PASCAL é usada como imagens de treinamento, a qual possui 30 vezes mais pontos-chave de treinamento do que a base de dados CMU10, o valor de mPM continua a aumentar quando mais que $T = 50$ triangulações de Delaunay são utilizadas: o valor de mPM aumentou de 0.49 para 0.58, quando o número de triangulações de Delaunay empregadas aumentou de $T = 50$ para $T = 200$. Esse resultado demonstra que a abordagem proposta neste trabalho para selecionar triplas de pontos-chave na imagem de teste é capaz de gerar um conjunto diversificado de grafos-chave, já que o uso de uma maior quantidade de triangulações de Delaunay foi capaz de compensar, parcialmente, a perda de desempenho acarretada pelo emprego de uma quantidade muito maior de imagens de treinamento (sem que aumentasse a quantidade $L$ de comparações iniciais entre pontos-chave).

### 4.4.3 Quantidade de grafos-chave selecionados versus número de triangulações

A Figura 4.8 apresenta a quantidade média de triplas de pontos-chave (grafos-chave) selecionadas em cada imagem de teste versus a quantidade de triangulações de Delaunay $T$ que é utilizada. À medida que a quantidade de triangulações aumenta, também aumenta a quantidade de grafos-chave repetidos que são produzidos a cada triangulação (isto é, grafos-chave que já haviam sido produzidos

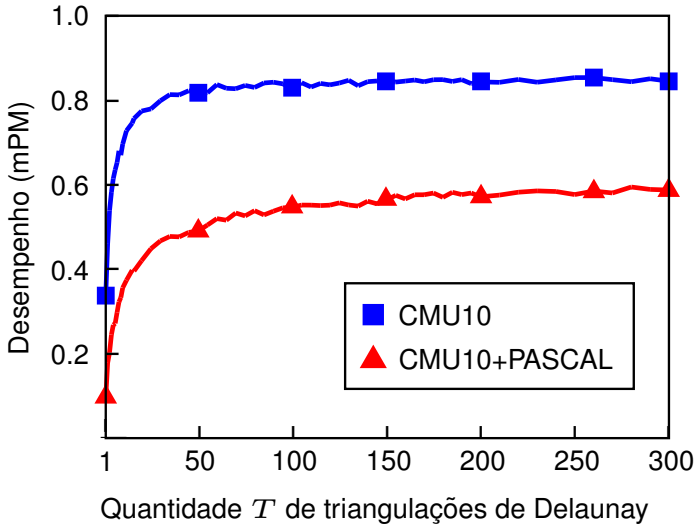


**Figura 4.7:** *Desempenho (média dos dez valores de Precisão Média, abreviado como mPM) versus o número $T$ de triangulações de Delaunay, com uso de imagens de treinamento obtidas das bases de dados CMU10 ou CMU10+PASCAL. Em ambos os casos, foi empregado um total de $L = 1000$ comparações entre cada ponto-chave da imagem de teste e pontos-chave de imagens de treinamento indexados.*

por uma triangulação realizada anteriormente). Assim, a quantidade total de grafos-chave produzida aumenta sub-linearmente com a quantidade de triangulações de Delaunay empregada.

### 4.4.4 Quantidade de correspondências entre grafos-chave

A Figura 4.9 apresenta a quantidade média de correspondências entre grafos-chave que ocorreram entre cada imagem de teste e o conjunto de todas as imagens de treinamento (como discutido na seção 3.3.2) versus o limiar utilizado para eliminar correspondências com uso da escala dos pontos-chave / comprimento das arestas de grafos-chave (equação 3.13). A eliminação de uma correspondência entre grafos-chave envolve o cálculo da razão entre a escala do ponto-chave e/ou o comprimento das arestas. No gráfico apresentado na Figura 4.9, foi variado o intervalo de valores de razão que permite que a correspondência entre grafos-chave seja considerada válida; na Equação 3.13, esse intervalo de valores de razão é $[0.5, 2]$. Quando um largo intervalo de $[1/60, 60]$ foi utilizado, virtualmente todas as correspondências entre grafos-chave foram aceitas; assim, a eliminação de correspondências entre grafos-chave foi feita somente pelo requisito de sentido de ciclo (anti-horário) e com uso da eliminação de correspondências entre grafos-chave baseada na orientação dos pontos-chave e arestas dos grafos-chave (a qual usa um liminar de 60°, como apresentado na Equação 3.14). Aumentar o intervalo de razões aceitas de $[0.5, 2]$ para $[1/60, 60]$ aumentou significamente, em 100 vezes, o número de correspondências entre grafos-chave estabelecidas (isto é, aumentou o número de correspondências incorretas entre grafos-chave).

Nesse experimento, foi empregado um total de $L = 1000$ comparações entre cada ponto-chave da imagem de teste e pontos-chave de imagens de treinamento indexados. Com uso da base de dados CMU10 como imagens de treinamento, o número de triangulações de Delaunay realizadas é $T = 50$; já com uso da base de dados CMU10+PASCAL, $T = 200$ é utilizado.

A Figura 4.10 é semelhante à Figura 4.9, mas a Figura 4.10 considera a eliminação de corres-

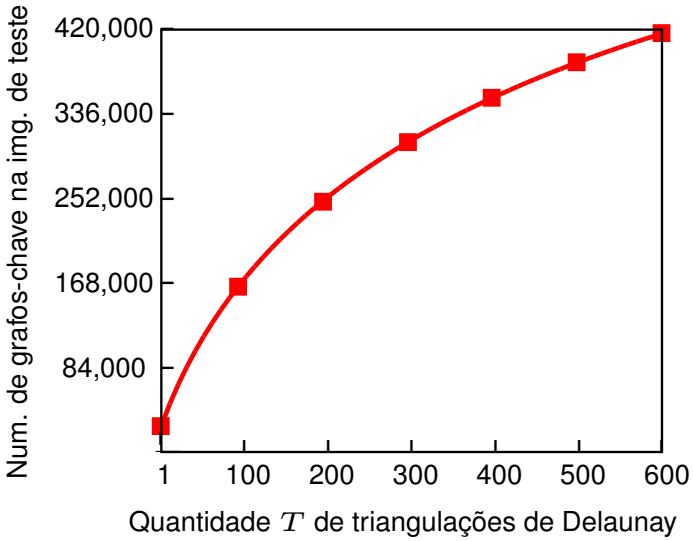


**Figura 4.8:** *Quantidade média de grafos-chave produzidos em cada imagem de teste versus quantidade $T$ de triangulações de Delaunay empregada.*

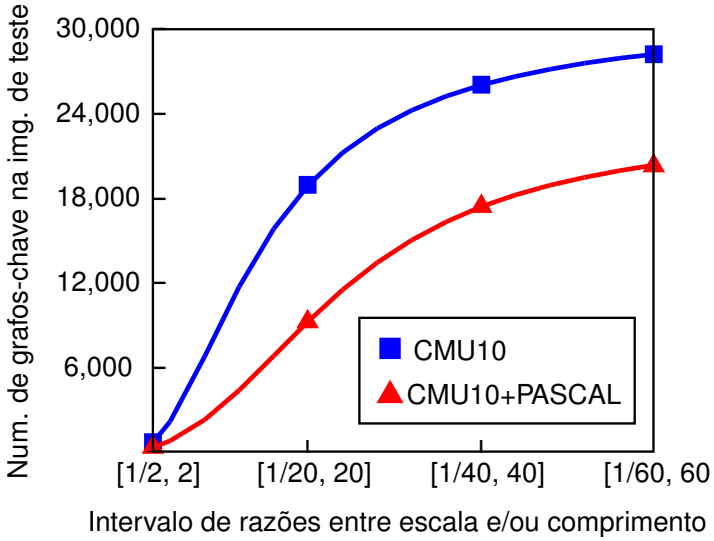


**Figura 4.9:** *Quantidade média de correspondências entre grafos-chave (em cada imagem de teste) versus intervalo de valores de razão (entre a escala dos pontos-chave e/ou o comprimento das arestas) em que as correspondências entre grafos-chave são consideradas válidas. Foi empregado um total de $L = 1000$ comparações entre cada ponto-chave da imagem de teste e pontos-chave de imagens de treinamento indexados. Com uso da base de dados CMU10 como imagens de treinamento, o número de triangulações de Delaunay realizadas é $T = 50$; já com uso da base de dados CMU10+PASCAL, $T = 200$ é utilizado.*

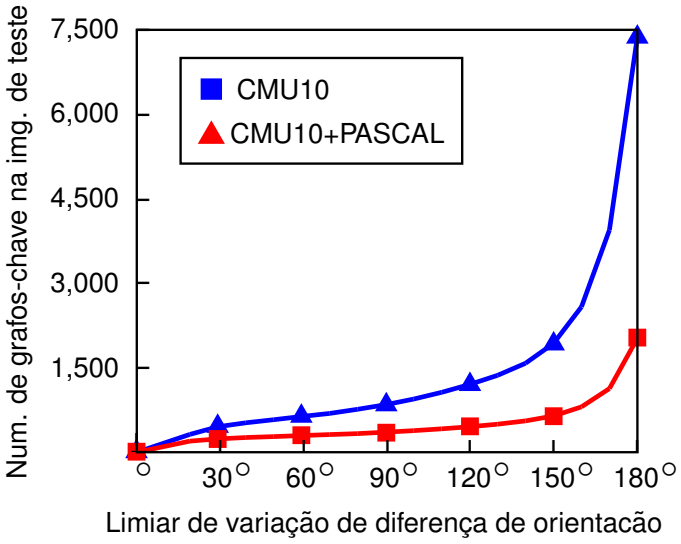


**Figura 4.10:** *Quantidade média de correspondências entre grafos-chave (em cada imagem de teste) versus a máxima diferença de variação de orientação de forma que a correspondência entre grafos-chave seja considerada válida.*

pondências entre grafos-chave que é feita com uso da orientação dos pontos-chave e da orientação das arestas dos grafos-chave. Quando a máxima diferença de variação de orientação que é aceita aumentou de 60° (Equation 3.14) para 180°, o número de correspondências entre grafos-chave estabelecidas aumentou em 10 vezes (isto é, aumentou o número de correspondências incorretas entre grafos-chave); nesse caso, a eliminação de correspondências entre grafos-chave foi feita somente pelo requisito de sentido de ciclo (anti-horário) e com uso da eliminação de correspondências entre grafos-chave baseada na escala dos pontos-chave e/ou comprimento das arestas (a qual usa um intervalo de $[0.5, 2]$ de razões aceitas, como apresentado na Equação 3.13).

Com o uso de ambas as equações 3.13 e 3.14 para realizar a eliminação de correspondências entre grafos-chave (além do sentido de ciclo), o número médio de correspondências entre grafos-chave estabelecido entre cada imagem de teste e o conjunto de todas as imagens de treinamento foi de 600, com o emprego da base de dados CMU10 como imagens de treinamento (com uso de $T = 50$ triangulações de Delaunay), e 400, com o emprego da base de dados CMU10+PASCAL como imagens de treinamento (com uso de $T = 200$ triangulações de Delaunay).

### 4.4.5 Quantidade máxima de poses de objeto avaliadas

A Figura 4.11 apresenta o desempenho versus a quantidade máxima de poses de objeto (transformações afim) que podem ser avaliadas, na etapa final do processo de reconhecimento de objetos, discutida na Seção 3.4.1. Somente a base de imagens de treinamento CMU10+PASCAL é empregada nesse experimento. Os resultados experimentais mostram que o método dos grafos-chave alcançou um melhor desempenho em comparação ao método SCRAMSAC e também em comparação ao método de Lowe (2004), mesmo quando era permitido que somente uma pequena quantidade de poses de objeto fossem avaliadas. Foi empregado um total de $L = 1000$ comparações entre cada ponto-chave da imagem de teste e pontos-chave de imagens de treinamento indexados; para o método

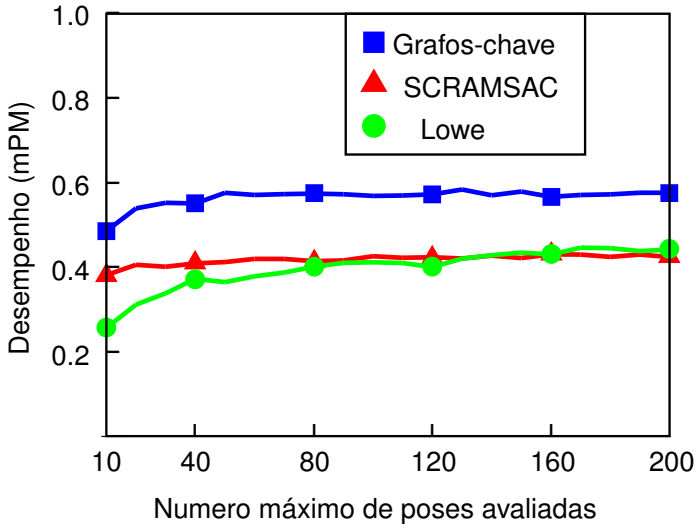


**Figura 4.11:** *Desempenho (média dos dez valores de Precisão Média, abreviado como mPM) versus quantidade máxima de poses de objeto que podiam ser avaliadas. Somente a base de dados CMU10+PASCAL foi utilizada como imagens de treinamento. Foi empregado um total de $L = 1000$ comparações entre cada ponto-chave da imagem de teste e pontos-chave de imagens de treinamento indexados; para o método dos grafos-chave, $T = 200$ triangulações de Delaunay foram empregadas.*

dos grafos-chave, $T = 200$ triangulações de Delaunay foram empregadas.

### 4.4.6 Quantidade máxima de correspondências entre pontos-chave no método SCRAMSAC

A Figure 4.12 apresenta o desempenho atingido pelo método SCRAMSAC versus o número máximo $N$ de correspondências que podem ser estabelecidas entre cada ponto-chave da imagem de teste e pontos-chave nas imagens de treinamento (como explicado na Seção 4.3). Dois casos foram avaliados: no primeiro caso, uma quantidade máxima de 1000 poses de objeto candidatas podiam ser avaliadas (de acordo com a discussão na Seção 3.4.1); no segundo caso, uma quantidade máxima de 10000 poses de objeto candidatas podiam ser avaliadas. Foi empregado um total de $L = 1000$ comparações entre cada ponto-chave da imagem de teste e pontos-chave de imagens de treinamento indexados.

Com uso da base de dados CMU10 como imagens de treinamento, os melhores resultados foram obtidos com uso de $N = 10$ (aproximadamente), enquanto que, com o uso da base de dados CMU10+PASCAL como imagens de treinamento, os melhores resultados foram obtidos com uso de $N = 50$. Em ambos os casos, o emprego de uma maior quantidade máxima de poses de objeto candidatas possibilitou que um melhor desempenho fosse alcançado.

Uma vantagem do método dos grafos-chave em relação ao método SCRAMSAC é que o método dos grafos-chave não possui um parâmetro similar a $N$, o qual limita a quantidade máxima de correspondências entre pontos-chave que podem ser inicialmente estabelecidas. Isto é, o método dos grafos-chave permite que a maior quantidade possível de correspondências entre pontos-chave seja estabelecida na etapa inicial; posteriormente, com emprego de informações estruturais de grafos-chave, a maior parte dessas correspondências entre pontos-chave iniciais será eliminada.

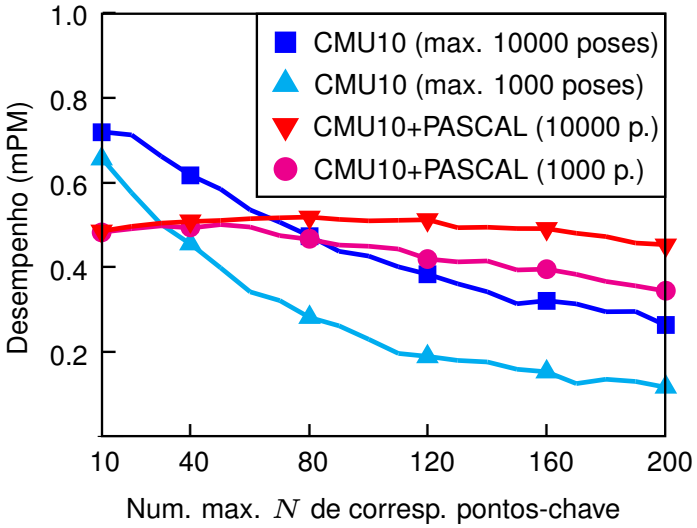


**Figura 4.12:** *Desempenho (média dos dez valores de Precisão Média, abreviado como mPM) alcançado pelo método SCRAMSAC versus a quantidade máxima $N$ de correspondências que cada ponto-chave na imagem de teste pode estabelecer com pontos-chave de imagens de treinamento. Foi empregado um total de $L = 1000$ comparações entre cada ponto-chave da imagem de teste e pontos-chave de imagens de treinamento indexados.*

### 4.4.7 Quantidade de comparações entre cada ponto-chave da imagem de teste e pontos-chave de imagens de treinamento indexados

A Figura 4.13 apresenta o desempenho versus a quantidade $L$ de comparações que são feitas entre um ponto-chave da imagem de teste e pontos-chave de imagens de treinamento, durante a etapa inicial que estabelece correspondências entre pontos-chave com uso da árvore hierárquica $k$-médias (explicada na Seção 3.2.2); a base de dados CMU10 foi utilizada como imagens de treinamento. Tanto para o método SCRAMSAC quanto para o método dos grafos-chave (com uso de $T = 50$), bons resultados foram obtidos com uso de $L = 1000$ comparações entre pontos-chave (ou seja, 0.4% do número total de pontos-chave de treinamento indexados). Já para o método de Lowe (2004), foi observado que o emprego de $L$ acima de 400 não proporcionou ganho significativo de desempenho.

A Figura 4.14 é semelhante à Figura 4.13, mas emprega a base de dados CMU10+PASCAL como imagens de treinamento. Tanto para o método dos grafos-chave (com uso de $T = 200$) quanto para o método SCRAMSAC, o uso de $L = 30000$ comparações entre pontos-chave permitiu que o mesmo desempenho alcançado com o uso da base de dados CMU10 como imagens de treinamento (e $L = 1000$) fosse alcançado com o uso da base de dados CMU10+PASCAL como imagens de treinamento. Esse é um resultado interessante, já que a quantidade de pontos-chave também cresceu 30 vezes, da base de dados CMU10 para a base de dados CMU10+PASCAL; isto é, o aumento da quantidade de comparações entre pontos-chave foi capaz de compensar o aumento no número de pontos-chave de imagens de treinamento armazenados. Já o método de Lowe (2004) não foi capaz de compensar, similarmente, o aumento no número de imagens de treinamento com o aumento no número de comparações entre pontos-chave, o que evidencia as limitações dessa abordagem em um cenário escalável.

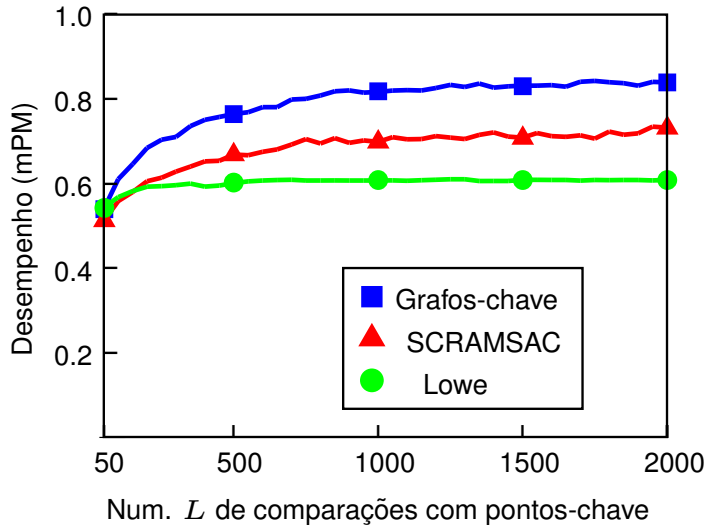


**Figura 4.13:** *Desempenho (média dos dez valores de Precisão Média, abreviado como mPM) versus o número $L$ de comparações entre cada ponto-chave da imagem de teste e pontos-chave de imagens de treinamento indexados, para a base de dados CMU10 sendo usada como imagens de treinamento. O método dos grafos-chave empregou $T = 50$ triangulações de Delaunay.*

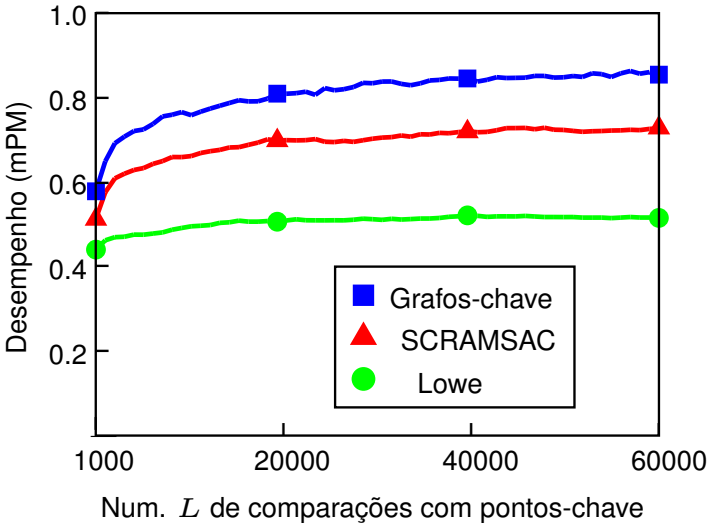


**Figura 4.14:** *Desempenho (média dos dez valores de Precisão Média, abreviado como mPM) versus o número $L$ de comparações entre cada ponto-chave da imagem de teste e pontos-chave de imagens de treinamento indexados, para a base de dados CMU10+PASCAL sendo usada como imagens de treinamento. O método dos grafos-chave empregou $T = 200$ triangulações de Delaunay.*

### 4.4.8 Eficiência do método proposto

O tempo computacional demandado pelo método dos grafos-chave foi mensurado com emprego de uma implementação na linguagem C. Com uso de $L = 1000$ comparações entre cada ponto-chave da imagem de teste e pontos-chave de imagens de treinamento indexados e $T = 50$ triangulações de Delaunay, independentemente de qual base de dados foi utilizada como imagens de treinamento (CMU10 ou CMU10+PASCAL), o tempo necessário para realizar o processo de reconhecimento de objetos em cada imagem de teste foi de $1.4 \pm 0.3$ segundos, em que 55% do tempo foi empregado no estágio inicial que estabelece correspondências entre pontos-chave individuais (com uso da árvore hierárquica $k$-médias), 3% do tempo foi empregado para selecionar os grafos-chave na imagem de teste (usando as $T = 50$ triangulações) e 39% do tempo foi empregado para estabelecer correspondências entre grafo-chave (no segundo estágio do processo de reconhecimento de objetos, explicado na seção 3.3.2). O tempo necessário para extrair características SIFT não foi considerado.

Com emprego da base de dados CMU10+PASCAL como imagens de treinamento e com uso dos parâmetros $L = 1000$ e $T = 200$, o tempo necessário para realizar o processo de reconhecimento de objetos em cada imagem de teste foi de $2.5 \pm 0.7$ segundos, em que 33% do tempo foi empregado no estágio inicial que estabelece correspondências entre pontos-chave individuais, 9% do tempo foi empregado para selecionar os grafos-chave na imagem de teste (com uso das $T = 200$ triangulações) e 57% do tempo foi empregado para estabelecer correspondências entre grafos-chave. Já se $L = 2000$ é usado (e $T = 200$), o tempo necessário, por imagem de teste, foi de $6.8 \pm 1.7$ segundos, em que o uso do tempo se dividiu entre 26% para estabelecer correspondências inicias entre pontos-chave, 3% para selecionar triplas na imagem de teste e 70% para estabelecer correspondências entre grafo-chave.

Em todos os experimentos reportados nesta seção, a etapa final do processo de reconhecimento de objetos (em que a pose do objeto é estimada, como discutido na Seção 3.4) utilizou aproximadamente 1% do tempo.

### 4.4.9 Sumarização dos resultados

Quando a base de dados CMU10 foi utilizada como imagens de treinamento, o desempenho alcançado pelo método dos grafos-chave, pelo método SCRAMSAC e pelo método de Lowe (2004) foram, respectivamente, 0.82, 0.70 e 0.61 (com uso de $L = 1000$ comparações entre cada ponto-chave da imagem de teste e pontos-chave de imagens de treinamento indexados, enquanto o método dos grafos-chave usa $T = 50$ triangulações de Delaunay). Já quando a base de dados CMU10+PASCAL foi empregada como imagens de treinamento (em que o método dos grafos-chave empregou $T = 200$ triangulações de Delaunay), o desempenho alcançado foi: com o uso de $L = 1000$: 0.58, 0.51 e 0.44; com o uso de $L = 2000$: 0.65, 0.58 e 0.46; e com o uso de $L = 30000$: 0.84, 0.70 e 0.51.

### 4.4.10 Comparação com os resultados reportados por Hsiao *et al.* (2010)

O desempenho (média dos dez valores de Precisão Média, abreviado como mPM) alcançado pelo método dos grafos-chave foi comparado ao valor de mPM reportado por Hsiao *et al.* (2010), que também empregou a base de dados CMU10 como imagens de treinamento. O valor de mPM alcançado pelo método dos grafos-chave, 0.82, foi superior ao valor de mPM reportado por Hsiao *et al.* (2010), 0.78, embora Hsiao *et al.* (2010) tenha empregado modelos tridimensionais de objetos de treinamento, o que possibilita que correspondências com pontos-chave em diferentes imagens de

treinamento sejam utilizadas simultaneamente, além de possibilitar que a instanciação do objeto encontrado na imagem de teste seja feita com uso do modelo tridimensional, o que potencialmente aumenta a quantidade de correspondências entre pontos-chave que "concordarm" com a pose de objeto instanciada (em comparação ao emprego de uma simples transformação afim entre imagens).

# Capítulo 5

# Conclusão

## 5.1 Considerações finais

Os grafos-chave propostos nesta tese consistem em um nível superior de características locais, acima do nível dos pontos-chave. O método dos grafos-chave constitui-se em uma estratégia para lidar com o grande número de correspondências entre pontos-chave que são estabelecidas com o uso de uma árvore de indexação de pontos-chave de imagens de treinamento, a qual produz, invariavelmente, uma grande maioria de correspondências falsas entre pontos-chave. A ideia é transformar essas correspondências entre pontos-chave em correspondências entre grafos-chave, em que um grafo-chave consiste em uma tripla de pontos-chave. Isso elimina, corretamente, a maior parte das correspondências iniciais entre pontos-chave, o que melhora tanto o desempenho quanto a eficiência do reconhecimento de objetos.

Esta tese apresenta duas contribuições principais. A primeira dessas contribuições consiste em várias propriedades estruturais que são usadas para verificar se cada correspondência entre grafos-chave é válida. Para que uma correspondência entre grafos-chave seja válida, é necessário que haja consistência entre o grafo-chave na imagem de teste e o grafo-chave na imagem de treinamento. As propriedades estruturais de grafos-chave englobam informações heterogêneas: o sentido de ciclo das triplas de pontos-chave, a escala, a orientação e a posição dos pontos-chave SIFT e o comprimento e a orientação das arestas dos grafos-chave.

A outra contribuição principal desta tese é um algoritmo eficiente para selecionar triplas de pontos-chave na imagem de teste, a partir dos pontos-chave inicialmente extraídos. Esse algoritmo de seleção de triplas baseia-se no uso de triangulações de Delaunay complementares. As triplas de pontos-chave geradas possuem propriedades favoráveis ao estabelecimento de correspondências: os pontos-chave que compõem uma tripla estão próximos entre si na imagem, sem que as arestas dos grafos-chave gerados sejam demasiado curtas.

Os resultados experimentais mostram que o tempo de execução do método dos grafos-chave depende, quase exclusivamente, de dois parâmetros, a quantidade $L$ de comparações entre pontos-chave feitas inicialmente e a quantidade $T$ de triangulações de Delaunay empregadas para selecionar triplas de pontos-chave na imagem de teste; de fato, um aumento de 30 vezes no número total de pontos-chave de imagens de treinamento, sem que os valores de $L$ e $T$ tenham mudado, não elevou o tempo de processamento. Além disso, o desempenho do método dos grafos-chave também não foi significativamente prejudicado por esse aumento no número de imagens de treinamento. Os resultados experimentais também mostram que, em comparação a outros métodos de reconhecimento de

objetos no estado-da-arte, o método dos grafos-chave alcançou um desempenho significativamente superior.

## 5.2 Sugestões para pesquisas futuras

Há duas direções de pesquisas futuras que são promissoras para melhorar o desempenho do método dos grafos-chave proposto nesta tese. A primeira refere-se ao emprego de modelos tridimensionais de objetos. Atualmente, objetos encontrados na imagem de teste são representados mediante o uso de simples transformações afim, que mapeiam a imagem de treinamento na imagem de teste. Contudo, quando há uma grande variação no ponto-de-vista entre o objeto na imagem de teste e o objeto na imagem de treinamento, transformações afim falham em representar, de forma satisfatória, o aspecto global do objeto. Além disso, modelos tridimensionais de objetos consistem em uma abordagem natural para combinar as correspondências entre pontos-chave que ocorrem entre a imagem de teste e diferentes imagens de treinamento.

Uma segunda abordagem para melhorar o desempenho do método dos grafos-chave está relacionada à estrutura de dados que é empregada para indexar os descritores de pontos-chave de imagens de treinamento. Estratégias mais elaboradas de indexação dos descritores têm um grande potencial de aperfeiçoar o método dos grafos-chave, já que a etapa inicial que estabelece correspondências entre pontos-chave é fundamental, demandando grande parte do tempo computacional e tendo influência decisiva no desempenho final do método. Nesse contexto, abordagens promissoras envolvem o uso de estruturas de dados mais complexas, por exemplo, o uso de múltiplas (e complementares) árvores de indexação. Ainda, métodos melhores para detector pontos-chave e extrair seus descritores podem ser empregados, por exemplo, os descritores compostos de variáveis binárias introduzidos por Simonyan *et al.* (2014), os quais empregam técnicas de aprendizado para mapear os descritores SIFT do espaço de descritores original para um espaço modificado, que possibilita um melhor desempenho na comparação entre os pontos-chave.

# Apêndice A

# Artigos publicados

Este anexo traz o artigos publicados durante o doutoramento. O primeiro artigo apresentado foi:

Estephan Dazzi, Teofilo de Campos, e Roberto M. Cesar Jr. “Improved object matching using structural relations.” Structural, Syntactic, and Statistical Pattern Recognition. Springer Berlin Heidelberg, 2014. 444-453.

O segundo artigo foi enviado para uma conferência e, durante a escrita desta tese, a resposta acerca da aceitação ainda não havia sido disponibilizada.

Ambos os artigos seguem a partir da próxima página.

# Improved object matching using structural relations

Estephan Dazzi[1,2⋆], Teofilo de Campos[2], and Roberto M. Cesar Jr.[1]

[1]Instituto de Matemática e Estatística - IME,
Universidade de São Paulo - USP, São Paulo, Brasil.
[2]CVSSP, University of Surrey, Guildford, GU2 7XH, UK.
{estephan.dazzi,roberto.cesar}@vision.ime.usp.br
t.decampos@st-annes.oxon.org

**Abstract.** This paper presents a method for object matching that uses local graphs called keygraphs instead of simple keypoints. A novel method to compare keygraphs was proposed in order to exploit their local structural information, producing better local matches. This speeds up an object matching pipeline, particularly using RANSAC, because each keygraph match contains enough information to produce a pose hypothesis, significantly reducing the number of local matches required for object matching and pose estimation. The experimental results show that a higher accuracy was achieved with this approach.

**Key words:** Local feature matching, SIFT, hierarchical k-means tree, RANSAC, graph-based structural information

## 1 Introduction

An important problem in computer vision is object recognition through matching, which consists in localising objects in test images and estimating their 3D pose. Solutions to this problem are useful in different application domains, such as robotics, medical images analysis and augmented reality. One of the most succesful approaches for this problem involves establishing correspondences between interest points (keypoints) in test and training images; next, a pose estimation algorithm is used, e.g. based on RANSAC, which operates by removing outliers that do not conform with global pose parameters. In this paper, we present a novel method that is capable of providing more accurate solutions besides being computationally cheaper. Our method is generic, in the sense that it can be used with any keypoint extractor method; we validate it using SIFT features [1].

In keypoint-based object recognition, point-to-point correspondences are obtained by matching discriminative features and reducing the set of matches in a

⋆ **Pre-print submitted to S+SSPR, Joensuu, Finland, 20-22 August 2014.** We would like to thank FAPESP (grant 2011/50761-2), CNPq, CAPES (process 14745/13-5) and NAP eScience - PRP - USP for the support. During most of the production of this paper, T. de Campos had been working in Neil Lawrence's group at the DCS, University of Sheffield, 211 Portobello, Sheffield, S1 4DP, UK.

post-processing step. For instance, the ratio test [1] compares the distances to the first and the second nearest neighbor and only establishes a match if the former is significantly smaller than the latter. However, usually there are locations on manmade objects that have similar local appearances, thus discriminative matching may prevent features with similar descriptors from being matched, which becomes a problem particularly when matching keypoints coming from different images.

In our work, initially, we also produce matches solely based on photometric information, but we allow a large number of matches to be established. Then, aiming to eliminate most of the incorrect matches, we use *structural information* within the images; this is done by establishing matches between small sets of keypoints, which we treat as graphs. In this way, our method produces a better set of keypoint matches, even when there are locations on the objects with similar appearances. Not only those matches have a high probability of being correct, but this also benefits the next stage, based on RANSAC, which ends up using a small set of graph correspondences.

Differently from previous approaches that model an image as a global graph and then proceed by employing graph matching methods, such as the work of Sirmacek and Unsalan [2] which uses SIFT keypoints as graph vertices or the work of McAuley and Caetano [3], our approach is local: we decompose the scene and pattern into collections of local graphs and perform only local graph matching, leaving the global matching to the RANSAC procedure. We built upon insights from the work of Morimitsu et al. [4], which focused on fast object detection using a single training image. For that, they used a graph edge descriptor based on Fourier transform and explicitly stored several structures obtained from the training image, which are matched to similar structures found in the test image. In the present paper, we focus on an object recognition task in which there are several images per training object and also many objects stored. We use a more discriminative keypoint extractor (SIFT), and since it is not computationally feasible to explicitly store structures found in the (many) training images, we develop a strategy based on quickly evaluating, during execution time, different aspects of strucures within test and training images.

## 2 Methodology

The first step of our object recognition process involves extracting SIFT keypoints from all the training images. We use the ground-truth segmentation to eliminate keypoints that are not on the object. We store all the training keypoints in a global indexing structure, which allows to quickly find the approximate nearest neighbors of a query (test) keypoint. We chose to use the hierarchical k-means tree proposed by Muja and Lowe [5] due to its efficiency. For each SIFT keypoint extracted from a training image we store its normalized descriptor (a 128-D feature vector), its scale, its orientation, an identifier of its source image and its $x$, $y$ position in that image.

Matching of a test object is done following a pipeline of three stages. First, photometric information is used: each SIFT descriptor of the test image runs through the hierarchical k-means tree, producing many matches to the keypoints of the training images. In the second stage, most of the incorrect keypoint matches are eliminated using structural information within images. The strategy consists in substituting the matches previously established between one-to-one keypoints by matches established between small sets of keypoints, i.e., graphs, called *keygraphs* [4]. A keygraph is a graph whose vertices are keypoints, and whose edges carry structural information about its keypoints. The third stage of the matching process consists in using a modified RANSAC (Random Sample Consensus) algorithm, which employs matches established between keygraphs.

### 2.1 Keypoint matching

SIFT keypoints are often located very close to each other and this can lead to poor pose estimation results with minimal sets. We select a maximal subset $\mathcal{S}$ of keypoints in the test image such that the distance, in pixels, between any two keypoints in $\mathcal{S}$ is above a threshold $d_{pix}$; we use $d_{pix} = 10$ pixels.

After selecting the set $\mathcal{S}$ of keypoints in the test image, we match them to the keypoints of the training images, which are stored in a hierarchical k-means tree. We let each test keypoint establish a match with at most *two* keypoints of *each* training image. In order to establish a match between keypoints, it is necessary that the Euclidean distance between their (normalized) SIFT descriptors is below a threshold $t$; we set $t$ with a relatively high value, as the next stage eliminates possibly incorrect matches. If a test keypoint can establish more than two matches with a same training image, only the two closest matches are kept.

### 2.2 Keygraph matching

A *keygraph* is defined as a graph $G = (V, E)$, where the vertex set $V$ is composed of keypoints, and $E$ is the set of graph edges. All the keypoints in a keygraph are present in the same image. Every keygraph has the same number of vertices, $\kappa$, and it consists in an *oriented circuit in the clockwise direction*, $G = (v_1, v_2, \ldots, v_\kappa)$.

Each keygraph in the test image can establish matches with keygraphs in every training image. Let $G = (v_1, v_2, \ldots, v_\kappa)$ and $H = (w_1, w_2, \ldots, w_\kappa)$ be keygraphs in a test and in a training image, respectively. The existence of a match between $G$ and $H$, denoted as $\mathcal{M} = (G, H)$, implies $\kappa$ matches between the keypoints (vertices) of $G$ and $H$. For instance, $(G, H)$ may imply the set of keypoints matches $\mathcal{M} = \{(v_1, w_1), (v_2, w_2), \ldots, (v_\kappa, w_\kappa)\}$, i.e., it implies the occurrence of $\kappa$ matches between pairs of keypoints.

**Obtaining keygraphs in the test image** We begin with the subset $\mathcal{S}$ of keypoints in the test image and execute the Delaunay Triangulation, generating a set of triangles, i.e. we use keygraphs with $\kappa = 3$ vertices, $G = (v_1, v_2, v_3)$, represented as triangles whose edges are oriented in the clockwise direction.

**Obtaining keygraphs in the training images** The keygraphs in the training images are not obtained using the Delaunay Triangulation. Instead, we first calculate the potential keygraph matches that may occur from the test image to each training image. Then we analyse which of those potential keygraph matches imply a valid keygraph in the training image.

Let $G = (v_1, v_2, v_3)$ be a keygraph in a test image, obtained using the Delauney Triangulation. For each training image, we verify whether $G$ establishes keygraph matches with that image. As an illustration, consider the case in which every keypoint of $G$, $v_1$, $v_2$ and $v_3$, establishes two matches with keypoints of a same training image; then there are *eight* different possible matches between $G$ and keygraphs of that training image: choose one of the two matches of $v_1$ *and* choose one of the two matches of $v_2$ *and* choose one of the two matches of $v_3$. Considering that the keypoint matches are $(v_1, w_1)$, $(v_1, w_2)$, $(v_2, w_3)$, $(v_2, w_4)$, $(v_3, w_5)$ and $(v_3, w_6)$, at most eight sets of keypoint matches (i.e. keygraph matches) can be established:

$$\mathcal{M}_1 = \{(v_1, w_1), (v_2, w_3), (v_3, w_5)\},\ \mathcal{M}_2 = \{(v_1, w_1), (v_2, w_3), (v_3, w_6)\},$$
$$\mathcal{M}_3 = \{(v_1, w_1), (v_2, w_4), (v_3, w_5)\},\ \mathcal{M}_4 = \{(v_1, w_1), (v_2, w_4), (v_3, w_6)\},$$
$$\mathcal{M}_5 = \{(v_1, w_2), (v_2, w_3), (v_3, w_5)\},\ \mathcal{M}_6 = \{(v_1, w_2), (v_2, w_3), (v_3, w_6)\},$$
$$\mathcal{M}_7 = \{(v_1, w_2), (v_2, w_4), (v_3, w_5)\} \text{ and } \mathcal{M}_8 = \{(v_1, w_2), (v_2, w_4), (v_3, w_6)\}.$$

Each one of those keygraph matches requires the existence of a specific keygraph in the training image; for instance, $\mathcal{M}_1 = \{(v_1, w_1), (v_2, w_3), (v_3, w_5)\}$ requires $H_1 = (w_1, w_3, w_5)$ in the training image. As we assume that mirroring is not a possible distortion of the test image, the circuit of a keygraph $H$ in a training image must be oriented in the clockwise direction; if it is oriented in the counter-clockwise direction then $H$ and the tentative keygraph match involving $H$ are not accepted[1].The set of possible keygraph matches $\mathcal{M}_1$, $\mathcal{M}_2$, ..., $\mathcal{M}_8$ established by a test keygraph $G = (v_1, v_2, v_3)$ with a training image can also be reduced if $v_1$, $v_2$ or $v_3$ establish fewer keypoint matches with that image; naturally, this set becomes empty if $v_1$, $v_2$ or $v_3$ does not establish any match or none of the implied circuits is in the clockwise direction.

**Discarding keygraph matches using structural relations** After obtaining a set of (at most eight) tentative keygraph matches between a test keygraph $G$ and keygraphs in a training image, we use five additional tests aiming to eliminate incorrect keygraph matches. This is very effective: the total number of keygraph matches is reduced in orders of magnitude. The tests are based on photometric and structural information within the keygraphs.

To illustrate the tests, let $G = (v_1, v_2, v_3)$ be a keygraph in the test image and $\mathcal{M} = \{(v_1, w_1), (v_2, w_2), (v_3, w_3)\}$ be a tentative keygraph match implied by $G$ in a training image, such that $\mathcal{M}$ requires the existence of the keygraph $H = (w_1, w_2, w_3)$ in that training image.

[1] We assume that training objects are convex or that there is, in the training set, at least one viewpoint of the object where this assumption is valid.

The first test is based on the edges of a training keygraph. In $H = (w_1, w_2, w_3)$, there are three edges: $e^H_{1,2} = (w_1, w_2)$, $e^H_{2,3} = (w_2, w_3)$ and $e^H_{3,1} = (w_3, w_1)$, whose lengths in pixels in the training image are denoted as, respectively, $|e^H_{1,2}|$, $|e^H_{2,3}|$ and $|e^H_{3,1}|$ (an edge is a straight line connecting two vertices). This first test verifies whether the edges length respect a minimum and a maximum value: we use 10 and 100 pixels, i.e. this test verifies whether $10 \leq |e^H_{i,j}| \leq 100$. As the keygraphs in the test image in general have edges with a length equal to or slightly greater than $d_{pix} = 10$ pixels, this test allows the objects to appear in the test image considerably smaller than the (large) object in the training image.

The second test is based on the ratio of edges length in corresponding keygraphs. Considering the tentative match $\mathcal{M}$ between the keygraphs $G = (v_1, v_2, v_3)$ and $H = (w_1, w_2, w_3)$, the three ratios between the length of corresponding edges are $r_{ij} = |e^G_{i,j}|/|e^H_{i,j}|$. This test verifies whether the larger ratio, $r_{ij}$, is at most twice the smaller one, $r_{kl}$, i.e. $r_{ij} \leq 2r_{kl}$. This test still allows the occurrence of a large variation between the viewpoints of the object in the test and the training images, but since many training images are taken around an object, a very drastic viewpoint change is not allowed to occur.

The third test is based on the ratio of the scale of corresponding SIFT keypoints. The motivation of this third test is similar to that of the second one. In the test keygraph $G = (v_1, v_2, v_3)$, consider that the scale of the SIFT keypoint $v_1$ is $s^G_1$ and similarly we have the scales $s^G_2$ for $v_2$ and $s^G_3$. In a similar way, for the training keygraph $H = (w_1, w_2, w_3)$ we have the scales $s^H_1$, $s^H_2$ and $s^H_3$. Thus the three ratios between the scale of corresponding keypoints are $r_1 = s^G_1/s^H_1$, $r_2 = s^G_2/s^H_2$ and $r_3 = s^G_3/s^H_3$. Similarly to the second test, this third test verifies whether the larger ratio is at most twice the smaller one.

The fourth test is based on both edges and scales. It uses results calculated in the second and the third tests: the ratios between edges length $r_{12}$, $r_{23}$ and $r_{31}$ and the ratios between scales $r_1$, $r_2$ and $r_3$. Ideally, the value $E = r_{12} + r_{23} + r_{31}$ would be similar to the value $S = r_1 + r_2 + r_3$, as the change in the object size and viewpoint from the training image to the test image should impact similarly the edges length and the SIFT scale. However, as imprecisions can occur, we let the values $E$ and $S$ differ: this fourth test verifies whether the larger value is at most 50% greater than the smaller value, i.e., if $E > S$ this test verifies whether $E \leq 1.5S$ and if $S > E$ it verifies whether $S \leq 1.5E$.

The fifth test uses the orientation (angle) of SIFT keypoints. One of the three pairs of matched keypoints is selected, and the variation of angle between the test and the training keypoints is calculated; then, for the other two keypoint pairs, this variation is applied and it is verified whether the resulting angle is within a margin of error of 45 degrees from the original SIFT orientation. The test suceeds if both keypoint pairs agree with the angle variation implied by the first keypoint pair. If the test fails using a keypoint pair to calcule the angle variation, it can be evaluated again using the other two keypoint pairs to calculate the angle variation: it must suceed for at least one of the three pairs. For example, in the tentative match $\mathcal{M}$ of keygraphs $G = (v_1, v_2, v_3)$ and $H = (w_1, w_2, w_3)$, the pair $(v_1, w_1)$ is used to calculate the angle variation. The angle of $v_1$ is $0^\circ$ and the

angle of $w_1$ is 30°, i.e. from $v_1$ to $w_1$ occurs an increasing of 30°. Now, this angle variation (+30°) is verified with the other two pairs of keypoints. The angles of $v_2$ and $w_2$ are, respectively, 40° and 80°; applying the variation of +30°, we obtain that the angle of $w_2$ should be 70° (40° plus 30°), which is within the margin of error of 45°, as the true orientation of $w_2$, 80°, is just 10° above the 70° implied by the first keypoint pair. A similar verification is made by applying the variation of +30° to the pair $(v_3, w_3)$. The whole evaluation can also be made using the angle variation from $v_2$ to $w_2$ or the angle variation from $v_3$ to $w_3$. We use a large margin of error of 45 degrees which allows the occurrence of imprecisions but still avoids the establishment of absurd keygraph matches.

Figure 2.2 illustrates the establishment of keygraph correspondences.

### 2.3 Third stage: RANSAC on keygraphs

One keygraph match generates $\kappa = 3$ keypoint matches. In the experiments in this paper, we use an affine transformation to instantiate an object, thus *one* keygraph match is necessary to instantiate an affine transformation. Compared to the tradicional RANSAC approach, which would require the random selection of three independent keypoint matches, the keygraph method requires the verification of a smaller number of poses.

Let $\mathcal{G}$ be the set of all keygraph matches between the test image and a training image, $\mathcal{G} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{|\mathcal{G}|}\}$, in which $\mathcal{M}_i$ is a set of three keypoint matches; thus the set $\mathcal{P}$ of *keypoint* matches between those images is $\mathcal{P} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \ldots \cup \mathcal{M}_{|\mathcal{G}|}$. To evaluate the quality of an affine transformation which instantiates, in the test image, the object present in that training image, we count the number of keypoint matches that agree with it: for each keypoint in the training image, let $x, y$ be its position in the test image as established by the keypoint match, and let $x', y'$ be its position in the test image as predicted by the affine transformation under evaluation. If the distance between $x, y$ and $x', y'$ is below three pixels, we consider that this keypoint match agrees with the transformation. If at least six keypoint matches agree with a transformation (i.e. the three matches used to instantiate it plus three other matches), we consider that a correct pose of the object is found, and the algorithm returns this affine transformation. If more than one solution is found for a test image, the algorithm returns the one with more matches agreeing with it.

## 3 Experiments and results

In our experiments we use a challenging object recognition dataset which contains ten different types of common household objects. For each object type, there are 25 training images taken around the object and 50 test images in which the object appears in a cluttered, realistic scene (in half of them there is one object instance, in the other half, two instances). This dataset was produced and made available by Hsiao et al. [6]. The authors evaluated it in a 3D object recognition task, in which a 3D model was created for each training object. In

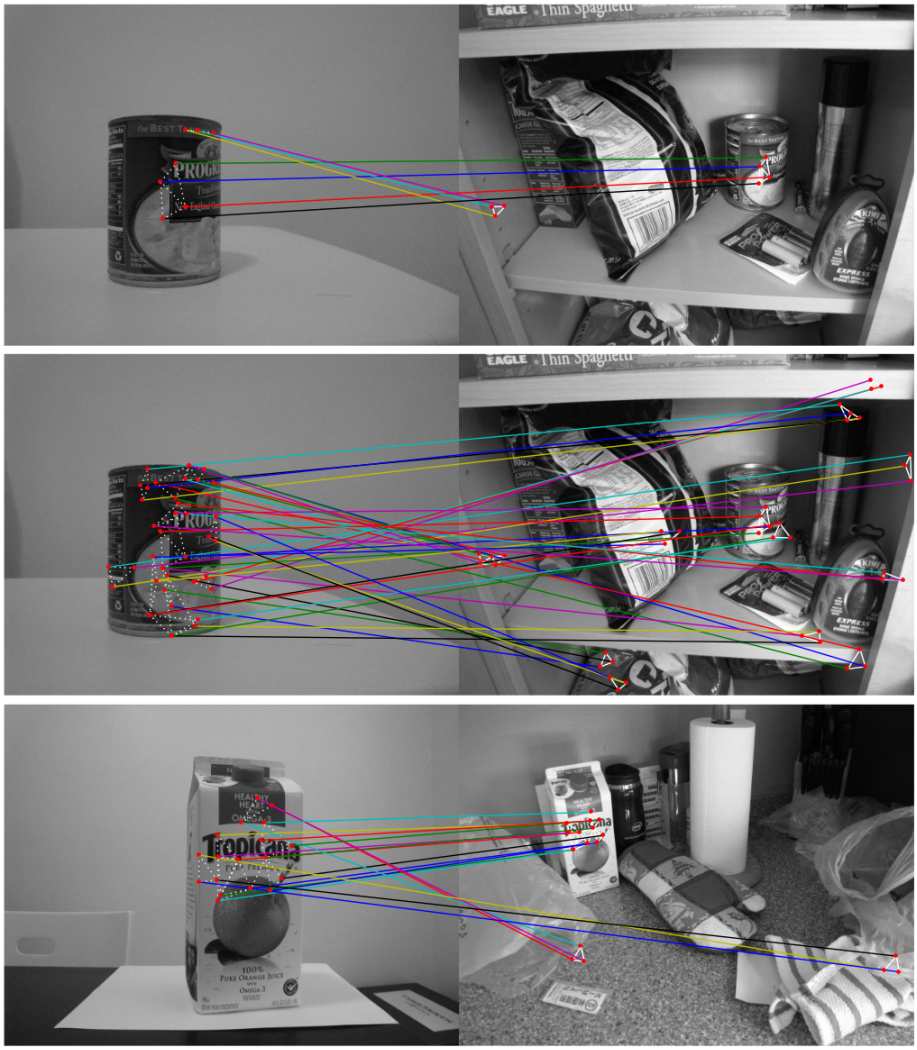


**Fig. 1.** Keygraph matches established between training (left side) and test (right side) images. Top: "clam chowder can" object: after using the five structural tests, three keygraph matches remain, where two of them are correct. Middle: for the same pair of images of the previous example, we show the keygraph matches that remain after using only four of the five tests; the fifth test, which uses SIFT angle, is not used. It can be seen that the number of (wrong) keygraph matches increases significantly. Bottom: "orange juice carton" object: after using the five tests, we obtain more correct keygraph matches than incorrect ones.

the present paper, we follow a simpler approach, in which we recognize objects by using an affine transformation between a test and a training image.

We compare the keygraph method proposed in this paper to the ratio test approach [1]. We run each SIFT descriptor $v$ of the test image through the hierarchical k-means tree [5] and obtain the nearest neighbor of $v$, which is at a distance $d_1$ from $v$, and the second nearest neighbor of $v$ that is of a *different* object type than the first nearest neighbor, which is at a distance $d_2$ from $v$; a match is established between $v$ and its nearest neighbor if $d_1/d_2 \leq 0.8$ [1]. Then, for every training image for which there are at least four keypoint matches to the test image, we use an exhaustive version of RANSAC, evaluating every possible combination of three keypoint matches to instantiate an affine transformation and then count the number of keypoint matches that agree with this pose. We consider that an instance of an object is found when at least four keypoint matches agree with a transformation (i.e. the three matches used to instantiate it plus one match). We use only four matches for the ratio test method, while for the keygraph method we use six matches (as explained in section 2.3), because the former usually produces fewer keypoints matches than the latter.

We use the same hierarchical k-means tree (with $k = 16$) for both methods, keygraph and ratio test. A query (test) keypoint is compared to a total of 4000 training keypoints stored in the tree leaves; the use of a smaller number of comparisons lowers the accurary of both methods. On average, a training image is described by 1080 SIFT keypoints (using the ground-truth segmentation) and a test image is described by 1070 keypoints (selected to compose the maximal subset $\mathcal{S}$ of keypoints). SIFT descriptors are normalized for zero mean and unit standard deviation; this normalization is useful because we use a threshold $t = 14$ for establishing keypoint matches in the first stage of the keygraph method.

We also compare our keygraph approach to the modified ratio test proposed by Hsiao et al. [6]. Aiming to establish more keypoint matches, the authors proposed to use the regular ratio test in conjunction with a modified ratio test which establishes discriminative matches with *clusters* of keypoints, such that establishing a match with a cluster produces matches to all the training keypoints in that cluster. For that, we create two additional hierarchical $k$-means trees (with $k = 16$ and $k = 32$). For each test keypoint, for each additional tree we verify whether that test keypoint establishes a discriminative match with a cluster composed of original training keypoints (i.e. a cluster that stores tree leaves); a discriminative match is established if $d_1/d_2 \leq 0.8$, in which $d_1$ is the distance to the closest cluster and $d_2$ is the distance to the second nearest cluster. We also use the traditional ratio test (using the original tree with $k = 16$), and employ all the established keypoint matches. Since the modified ratio test generates more keypoint matches then the ratio test, we consider that an instance of an object is found when at least six keypoint matches agree with a transformation, similarly to the keygraph method (for the ratio test, we consider that a pose is found when four keypoint matches agree it).

Table 3 summarizes the results obtained. When a pose is found, we manually verified it by checking if the correct viewpoint of the object was projected in

the test image. For the test images with two object instances, we consider that finding just one of them is a correct solution.

**Table 1.** Percentage of test images for which a correct object was found (and for which a wrong object was found), i.e. true positives (and false positives), for the original ratio test [1], the modified ratio test [6] and the keygraph method.

| Object type | Ratio test (Lowe [1]) | Modified ratio test (Hsiao et al. [6]) | Keygraph method (this paper) |
|---|---|---|---|
| Clam chowder can | 14% (4%) | 22% (4%) | 46% (2%) |
| Soy milk can | 2% (12%) | 2% (10%) | 8% (6%) |
| Tomato soup can | 14% | 10% (4%) | 36% |
| Orange juice carton | 54% (4%) | 58% (2%) | 72% |
| Soy milk carton | 44% (8%) | 46% (4%) | 54% (4%) |
| Diet coke can | 0% (2%) | 2% (6%) | 2% |
| Pot roast soup | 10% (2%) | 6% (4%) | 36% |
| Juice box | 26% (10%) | 32% (12%) | 42% (6%) |
| Rice pilaf box | 64% | 62% (2%) | 74% (2%) |
| Rice tuscan box | 68% (4%) | 58% | 62% (2%) |

Our method performs significantly better than the ratio test and the modified ratio test. In the matching stage (before RANSAC), the keygraph method established an average of 2.8 keygraph matches (7.4 keypoint matches) between a test image and each training image, while the modified ratio test, on average, established only 1.7 keypoint matches between a test and each training image; this number could be increased by using additional $k$-means trees in the modified ratio test, but we observed that this also increased the false positive rate.

Before the RANSAC stage, the computational time demanded by the ratio test method and the keygraph method is similar, as the time spent to establish keypoint matches through the hierarchical k-means tree is largely dominant in comparison to the next stage of keygraph matching.

## 4 Conclusion

In this paper we described a method for object matching based on keygraphs, rather than keypoints, i.e., objects are matched using sets of triangles, where each vertex is a keypoint detected and described using SIFT. In the first step, keypoints are matched using a hierarchical k-means tree. Delaunay triangulation generates keygraphs in the test image and the matched keypoints generate keygraphs in the training images. We proposed to use five triangle features in order to evaluate the match between keygraphs, removing a significant number of false matches before running RANSAC to select inliers to compute an affine transformation between training and test images. Our method achieved a significantly higher accuracy than two state-of-the-art methods, the ratio test [1]

and the modified ratio test [6]. Furthermore, the number of keygraph matches is small. On average, 2.8 keygraph matches are established between a test image and each training image. The quality of these matches is high, i.e., a small number of false matches occur. Besides, each keygraph match carries enough information to instantiate a pose hypothesis using an affine transformation. On the other hand, the ratio test method requires at least three keypoint matches.

As future work, we plan to use our method for 3D object recognition and pose estimation as in [6], which uses a structure-from-motion algorithm to create a 3D model of each training object. We believe that our approach is especially suited for this 3D setting. Only two keygraph matches between a test image and (possibly different) training images generate six keypoint matches, which is a good minimal set to generate a 3D pose [7]. This is an important advantage in comparison to a method that solely uses keypoints, which requires the selection of six keypoint matches to instantiate a 3D pose [7] or the selection of four keypoints matches with the use of an algorithm such as EPnP [8]. We expect that the keygraph method will demand a smaller number of 3D pose evaluations.

Another future work involves the use of Domain Adaptation (D.A.) techniques, which are useful when the training data is different from the test data (e.g. [9]). Such a domain change can occur due to variations in the object viewpoint, camera parameters, illumination change, motion etc. We expect that the use of D.A. will improve the first stage of our method (keypoint matching), as this stage is, essentially, a classification task through the hierarchical k-means tree. We also suggest the use of structured learning methods in order to optimize the weight of features used for graph matching, as done in [10].

## References

1. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.
2. B. Sirmacek and C. Unsalan. Urban-area and building detection using SIFT keypoints and graph theory. *IEEE Trans. on Geoscience and Remote Sensing*, 2009.
3. J. J. McAuley and T. S. Caetano. Fast matching of large point sets under occlusions. *Pattern Recognition*, 2012.
4. H. Morimitsu, M. Hashimoto, R. Pimentel, R. M. Cesar, and R. Hirata. Keygraphs for sign detection in indoor environments by mobile phones. In *8th IAPR-TC-15 Workshop on Graph-Based Representations in Pattern Recognition*, LNCS. 2011.
5. M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISSAPP*, 2009.
6. E. Hsiao, A. Collet, and M. Hebert. Making specific features less discriminative to improve point-based 3D object recognition. In *CVPR*, 2010.
7. R. Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.
8. V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate O(n) solution to the PnP problem. *IJCV*, 2009.
9. J. Ni, Q. Qiu, and R. Chellappa. Subspace interpolation via dictionary learning for unsupervised domain adaptation. In *CVPR*, 2013.
10. J. J. McAuley, T. de Campos, and T. S. Caetano. Unified graph matching in Euclidean spaces. In *CVPR*, 2010.

# Scalable object recognition using smart sampling of keypoint triples and keygraph structure

Anonymous ICCV submission

Paper ID 1408

## Abstract

*This paper presents an object recognition method that employs matches between local graphs of keypoints, called keygraphs, instead of simple keypoint matches. For a keygraph match to be valid, the vertices (keypoints) descriptors must be similar and both keygraphs must satisfy structural properties concerning keypoints orientation, scale, relative position and cyclic ordering. This allows to correctly filter out the large majority of the initial keypoint matches. This heterogeneous set of structural information for keypoint match filtering is our first main contribution. Our second main contribution is an algorithm to smartly sample triples of keypoints (*i.e. *keygraphs) in a query image, which is based on the use of complementary Delaunay triangulations. The algorithm generates a number of triples that grows linearly with the number of keypoints in the query image. Query keygraphs are then matched against the indexed training keypoints; each established keygraph match is used to evaluate a candidate pose (an affine transformation). The proposed method has been evaluated for object recognition and pose estimation, achieving a better performance in comparison to state-of-the-art methods.*

## 1. Introduction

Many problems in computer vision involve matching an image against a large database of images. For instance, the problem of 3D object recognition can be cast as one of image matching, by storing images of a range of views for each training object and then treating a test image as a query.

One of the most succesful approaches for image matching is the local feature framework [9], which extracts interest points (keypoints) from images. Currently, most methods on the literature follow the strategy of establishing one-to-one keypoint correspondences (*e.g.* [8, 13, 6, 12]), relying only on the similarity between keypoint descriptors.

However, establishing simple one-to-one keypoint correspondences disregards structural aspects contained in neighborhoods of keypoints in images. Recent works on the literature have attempted to use such a structural information (*e.g.* [4, 3, 13]), but in limited ways.

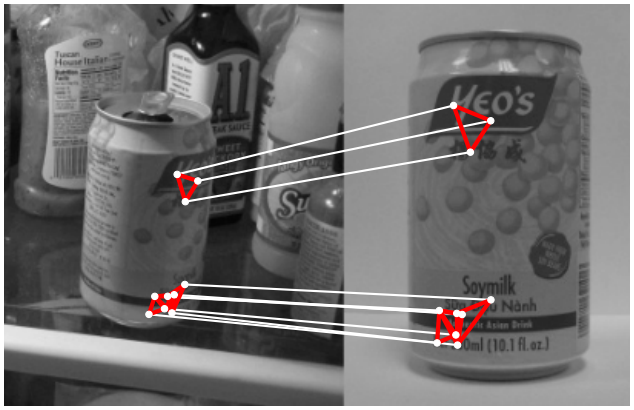


Figure 1. Keygraph (*i.e.* keypoint triple) matches between a query image (left) and a training image (right). Keygraphs are sampled in the query image by using complementary Delaunay triangulations and then matched against the training keypoints. A valid keygraph match must satisfy structural properties concerning keypoints orientation, scale, relative position and cyclic ordering.

This paper presents an improved approach to deal with structural organisations of keypoints in images. We substitute matches between keypoints by matches between *local graphs of keypoints*. Each image is represented by a set of local, small graphs, whose vertices are keypoints and whose edges contain structural information; these local keypoint graphs are called *keygraphs*. Correspondences are then established between keygraphs in the query image and keygraphs in the training images. Establishing a match between graphs encompasses not only keypoint descriptor information (vertex matches) but also requires both graphs to satisfy structural properties (see Figure 1).

Our first main contribution is a novel set of graph structural properties which are used to filter out candidate keygraph matches. Keygraph structure concerns keypoints orientation, scale, relative position and cyclic ordering. The filtering stage correctly rejects more than 99% of the ini-

tial candidate keygraph matches, which improves efficiency and performance in the final pose estimation stage.

Our second main contribution is an algorithm to smartly sample triples of keypoints (*i.e*. keygraphs) in a query image. Initially, we sample a subset of keypoints such that no two keypoints are too close. Then, a Delaunay triangulation is constructed, producing triples of neighbouring keypoints. This process is repeated $T$ times; keypoint subsets are selected in a pairwise complementary way. For a query image with $n$ keypoints, running the algorithm has complexity $O(Tn \log n)$ and produces $O(Tn)$ keypoint triples.

Keygraphs in the query image are matched against indexed [11] training keypoints; each established keygraph match generates a candidate object pose (an affine transformation). Thus the number of pose evaluations grows *linearly* with the number of keypoint matches between a pair of images, while RANSAC [2] evaluates a *polynomial* number of poses (*e.g*. cubic, for affine transformations).

Keygraphs constitute an intermediate level feature, above the keypoints. Keygraphs can be used with any keypoint detector that assigns scale and orientation to every keypoint. In this paper, we employ SIFT [8] features.

We evaluated our method for object recognition and localisation. A better performance was achieved in comparison to state-of-the-art methods [8, 13, 6], particularly in a scalable scenario, where many training images were used.

## 2. Related work

The concept of keygraphs was introduced by Hashimoto *et al*. [5] and used for indoors self-localisation using sign boards by Morimitsu *et al*. [10]. Fourier coefficients of keygraph edges were used as local descriptors, providing a high efficiency. However, in comparison to our method, Morimitsu *et al*. only employed cyclic ordering for match filtering and only used a single Delaunay triangulation to generate query keygraphs. Also, all the training keygraphs must be stored in memory during application time, which imposes a limit on the number of training images. On the other hand, we propose a scalable keygraph based method.

Philbin *et al*. [12] proposed an image retrieval method which uses affine-invariant keypoints [9]. This allows a single *keypoint* match to carry enough information to instantiate a candidate pose, while we use a single *keygraph* match for that. For object recognition, there are two advantages of keygraph matching over keypoint matching. First, generating more robust candidate poses, as three image regions are employed. Second, using structural information for match filtering, which is not possible if only one point is matched.

Hsiao *et al*. [6] presented a 3D object recognition method which uses 3D object models built using structure-from-motion. Keypoints were extracted from many artificially distorted training images and then triangulated into the 3D model, which augments the set of training keypoints.

| Info / Method | [13] | [4] | [3] | [7] | [10] | Ours |
|---|---|---|---|---|---|---|
| Cyclic ordering | | | X | | X | X |
| Keypoints scale | | X | X | X | | X |
| Kpt. orientation | | | | X | | X |
| Kpt. relative position | | X | X | | | X |
| Kpt. neighbourhood | X | X | | | X | X |

Table 1. Structural information employed in keypoint matching.

Sattler *et al*. [13] explored structural information in keypoint neighbourhoods. The idea is that a correct keypoint match usually occurs close to other correct keypoint matches in the image. Thus each keypoint match $(p, q)$ is required to have some neighbour matches around $p$ (in the query image) and around $q$ (in the training image) agreeing about the matched image, otherwise $(p, q)$ is filtered out.

We also explore the fact that a correct keypoint match is likely to be close to other correct keypoint matches in the image. A keypoint triple in the query image is composed of neighbour keypoints, which increases the chance that all the three keypoint matches are simultaneously correct.

Hao *et al*. [4] presented a 3D object recognition method. First, a filtering stage, similar to the method by Sattler *et al*. [13], is employed. Then, another structural property is evaluated: for every pair of matches, the method requires agreement between distances in the 3D model and keypoint scales. However, in comparison to our method, [4] neither employs cyclic ordering nor keypoint orientation for keypoint match filtering, which decreases efficiency.

Hao *et al*. [3] employed triples of keypoint matches, similarly to the proposed keygraphs method. For keypoint match filtering, the authors used structural information concerning keypoints scale, relative position and cyclic ordering. Before application time, the method in [3] explicitly stores a large number of training keypoint triples, which are then matched against the keypoints in a query image.

The method by Hao *et al*. [3] differs from ours in three main aspects. First, we explore the use of local neighborhood of keypoints. Second, our method additionally uses keypoints orientation for match filtering, which increases efficiency. Third, in [3], a large memory space is needed to store a sufficiently diverse set of training keypoint triples. On the other hand, in the proposed method, every keygraph is efficiently generated during execution time.

Jégou *et al*. [7] presented an image retrieval method which uses structural information for keypoint match filtering, by requiring consistency in keypoints scale and orientation. However, the method neither uses cyclic ordering nor verify the consistency of keypoints scale and orientation with their relative position, leading to a lower efficiency in keypoint match filtering. Also, the authors did not explore the idea of using local keypoint neighborhoods.

Table 2 summarizes the previous discussion concerning

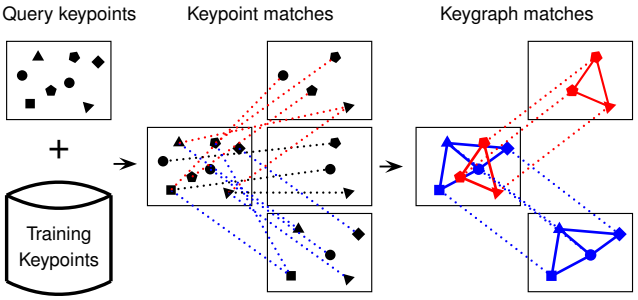


Figure 2. Establishing keygraph matches between a query image and training images. Query keypoints are matched against indexed training keypoints, establishing keypoint matches. Then, keypoint matches are transformed into keygraph matches, which correctly eliminates the large majority of the initial keypoint matches.

the structural information used for keypoint match filtering. Our method employs a more heterogeneous set of information, which provides a better performance in filtering.

## 3. Proposed method

Object recognition is carried out by finding affine transformations mapping the query image to one or more training images. Before application time, the training keypoints are extracted and indexed [11]. During application time, our method follows this pipeline:

1. *Keypoint matching*: Each keypoint in the query image runs through a hierarchical k-means tree [11], matching keypoints in many training images (see Figure 2).

2. *Keygraph matching*: Keypoint triples are sampled in the query image (see Figure 3) and then matched against the training images, by using the previously obtained keypoint matches (see Figure 2). This generates candidate keygraph matches. Each candidate keygraph match must satisfy the keygraph structural properties in order to be considered as valid (see Figure 4).

3. *Pose estimation*: The remaining keygraph matches are used to search for affine transformations mapping query and training images. Each keygraph match generates a candidate transformation that is evaluated by counting the number of keygraph vertex matches (*i.e.* filtered keypoint matches) that agree with it.

### 3.1. First stage: keypoint matching

The present work employs a hierarchical $k$-means tree [11] to index training keypoints. The tree is built by recursivally applying $k$-means to the descriptor space; each leaf node stores a set of (less than $k$) training keypoints.

During application time, each query keypoint $p$ runs through the tree in order to match stored training keypoints. $p$ is compared to a stored keypoint $q$ by calculating the Euclidean distance between their SIFT descriptors [8].

An important parameter is the number $L$ of keypoint comparisons that are done between a query keypoint $p$ and stored training keypoints. The key advantage of employing a keypoint indexing tree is that setting $L$ to a small value, such as $0.01\%$ of the total number of stored training keypoints, already provides good detection results.

The original $k$-means tree [11] finds the single nearest neighbour of a query keypoint $p$. We follow a different strategy, aiming to find many good matches. We let $p$ establish up to *one* match with a keypoint in *each* training image, namely the match with the lowest Euclidean distance.

### 3.2. Second stage: keygraph matching

In the second stage, keygraphs are obtained from the query image and then matched againt the training images.

A keygraph is defined as a graph $G = (V, E)$, where the vertex set $V$ is composed of keypoints extracted from the same image, and $E$ is the set of graph edges. Every keygraph $G$ has $\kappa$ vertices and consists is an *oriented circuit in the counter-clockwise direction*, $G = (v_1, v_2, \ldots, v_\kappa)$ [10].

We employ keygraphs with $\kappa = 3$ vertices, which allows the use of cyclic ordering and makes *one* keygraph match be sufficient to instantiate an affine transformation.

#### 3.2.1 Sampling keygraphs in a query image

Obtaining keygraphs from a query image involves sampling from the set of all possible triples of keypoints. RANSAC approaches this by independently selecting each keypoint match in a triple; however, this causes the number of evaluated affine transformations to increase *cubically* with the number of keypoint matches between a pair of images. Our method follows a more elaborate strategy, based on using complementary Delaunay triangulations, which produces a *linear* number of triples and affine transformations while each triple is a local neighbourhood of keypoints.

We avoid using keygraphs with very short edges, which would increase sensitivity to noise. We select a keypoint subset $\mathcal{S} \subseteq \mathcal{P}$ from the original set $\mathcal{P}$ of query keypoints such that no keypoints in $\mathcal{S}$ are very close to each other in the image. As $|\mathcal{S}|$ can possibly be much smaller than $|\mathcal{P}|$, we use $T$ different subsets, $\mathcal{S}_1, \ldots, \mathcal{S}_T$, virtually guaranteeing that every keypoint from $\mathcal{P}$ is selected at least once.

To select the first keypoint subset $\mathcal{S}_1 \subseteq \mathcal{P}$, start with $\mathcal{S}_1$ as an empty set. Then, keypoints $p_j$ are randomly accessed in $\mathcal{P}$, and a tentative is made to include each $p_j$ in the set $\mathcal{S}_1$ being constructed. The construction of $\mathcal{S}_1$ stops after all the keypoints in $\mathcal{P}$ have been accessed. To include a keypoint $p_j$ in $\mathcal{S}_1$, the $\ell_\infty$ distance, in pixels, between $p_j$ and every keypoint already included in $\mathcal{S}_1$ must be above $\delta = 8$ pixels.

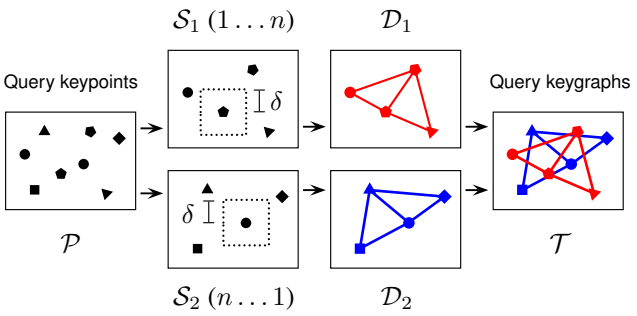


Figure 3. Sampling keypoint triples in a query image. First, query keypoints $\mathcal{P}$ are detected. Then, complementary keypoint subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ are obtained (using a minimum $\ell_\infty$ distance between keypoints of $\delta$ pixels) and employed to calculate Delaunay triangulations $\mathcal{D}_1$ and $\mathcal{D}_2$. The triples $\mathcal{T} = \mathcal{D}_1 \cup \mathcal{D}_2$ are returned.

Random selection of keypoints is implemented by creating an array $A$ of keypoints from $\mathcal{P}$, and then randomly permuting the elements in $A$. Then, for selecting the first keypoint subset $\mathcal{S}_1$, the sequence of accessed keypoints is obtained by starting from the *first* position of the array $A$ and then increasing one position at a time until the last element of the array is accessed. Each time an array position $j$ is accessed, the method tries to include the keypoint $p_j$ stored in this position into the set $\mathcal{S}_1$ being constructed.

After the creation of the first keypoint subset $\mathcal{S}_1 \subseteq \mathcal{P}$, the next subset $\mathcal{S}_2 \subseteq \mathcal{P}$ is constructed. The same array of permuted keypoints $A$ that was used for $\mathcal{S}_1$ is also used for $\mathcal{S}_2$, but, for $\mathcal{S}_2$, the elements of $A$ are accessed starting from the *last* position of the array and then decreasing one position at a time until the first position of $A$ is achieved.

The complexity of selecting a keypoint subset $\mathcal{S}_i$ is $O(n)$, where $n$ is the number of keypoints in the query image. Permuting the positions of a keypoint array $A$ has complexity $O(n)$. Then, keypoints from $A$ are included into $\mathcal{S}_i$; trying to include a keypoint has complexity $O(1)$, thanks to the use of a grid of squares with cell length $\delta = 8$ pixels[1].

To obtain $T$ keypoint subsets, the procedure described to obtain $\mathcal{S}_1$ and $\mathcal{S}_2$ is simply repeated. Each pair of subsets $\mathcal{S}_i$ and $\mathcal{S}_{i+1}$ employs the same array $A_i$ of permuted keypoints.

Finally, each keypoint subset $\mathcal{S}_i$ is used to generate a Delaunay triangulation. $\mathcal{S}_i$ produces the set of keypoint triples $\mathcal{D}_i = \{(p_1, p_2, p_3), (r_1, r_2, r_3), \ldots\}$, with each triple in $\mathcal{D}_i$ being a keygraph. Calculating the Delaunay triangulation $\mathcal{D}_i$ has complexity $O(n \log n)$ and produces $O(n)$ triples.

The set of all keypoint triples is given by $\mathcal{T} = \mathcal{D}_1 \cup \ldots \cup \mathcal{D}_T$. Every keypoint triple in $\mathcal{T}$ is included in a hash table for checking in $O(1)$ time whether that triple is repeated.

In summary, calculating the set $\mathcal{T}$ of keygraphs in a query image has complexity $O(Tn \log n)$. This produces a number of keygraphs (keypoint triples) $|\mathcal{T}| = O(Tn)$.

[1]Including a keypoint $p$ in a grid cell demands checking if this cell is already occupied by a keypoint and then checking the $\ell_\infty$ distance between $p$ and a (possible) keypoint in each of the eight neighbouring grid cells.

### 3.2.2 Matching query and training keygraphs

Keygraphs in the training images are not generated by a Delaunay triangulation. Instead, we first calculate potential keygraph matches that may occur, based on the initial keypoint matches. Then, the candidate keygraph matches are filtered out by using structural information.

**Obtaining initial candidate keygraph matches**. A keygraph in the query image can establish up to *one* match with each training image. Let $G = (v_1, v_2, v_3)$ be a query keygraph; $G$ can be matched in all images where all its vertices have reported correspondences. During the keypoint matching stage, assume that the keypoint matches $(v_1, u_1)$, $(v_2, u_2)$ and $(v_3, u_3)$ are established between the query image and a training image $I$. Thus the candidate keygraph match between $G$ and the image $I$ is $(G, H)$, where $H = (u_1, u_2, u_3)$ is a candidate keygraph in the training image $I$. Therefore, the keygraph match $(G, H)$ is associated to the keypoint matches $\mathcal{M} = \{(v_1, u_1), (v_2, u_2), (v_3, u_3)\}$.

Implementing this initial stage involves checking, for each query keygraph $G = (v_1, v_2, v_3)$, which training images have keypoint matches with $v_1$, $v_2$ and $v_3$. To calculate this, we use, for each query keypoint $p$, words of 64 bits indicating which training images are matched by $p$. Thus the images initially matched to $G = (v_1, v_2, v_3)$ are obtained by using a AND binary operation between the words for $v_1$, $v_2$ and $v_3$; the 1's in the resulting binary words can be accessed by using count-leading-zeros machine instructions.

**Keygraph match filtering using structural information**. Every keygraph match $(G, H)$ must satisfy the graph structural properties in order to be considered as a valid match.

*Cyclic ordering* is a topological property, invariant to perspective transformation in 2D images. Every keygraph $G$ in a query image is a circuit (a triple) oriented in the counter-clockwise direction. Since it is assumed that mirroring is not a possible distortion of the query image, the circuit of a matched keygraph $H$ in a training image has to be oriented in the counter-clockwise direction as well.

Four other attributes are calculated for each keygraph:

- *Edges length*: A keygraph edge is a straight line connecting two vertices with length expressed in pixels. A keygraph $U = (h_1, h_2, h_3)$ has three edges, $e_{12} = (h_1, h_2)$, $e_{23} = (h_2, h_3)$ and $e_{31} = (h_3, h_1)$, whose lengths are $l^U_{12}$, $l^U_{23}$ and $l^U_{31}$ (see Figure 4-a).
- *Keypoints scale*: Each keypoint in $U = (h_1, h_2, h_3)$ has a scale assigned by SIFT. They are denoted as $s^U_1$, $s^U_2$ and $s^U_3$, respectively (see Figure 4-b).
- *Edges orientation*: Each keygraph edge forms an angle with the $x$ axis of the image (a keygraph edge is treated

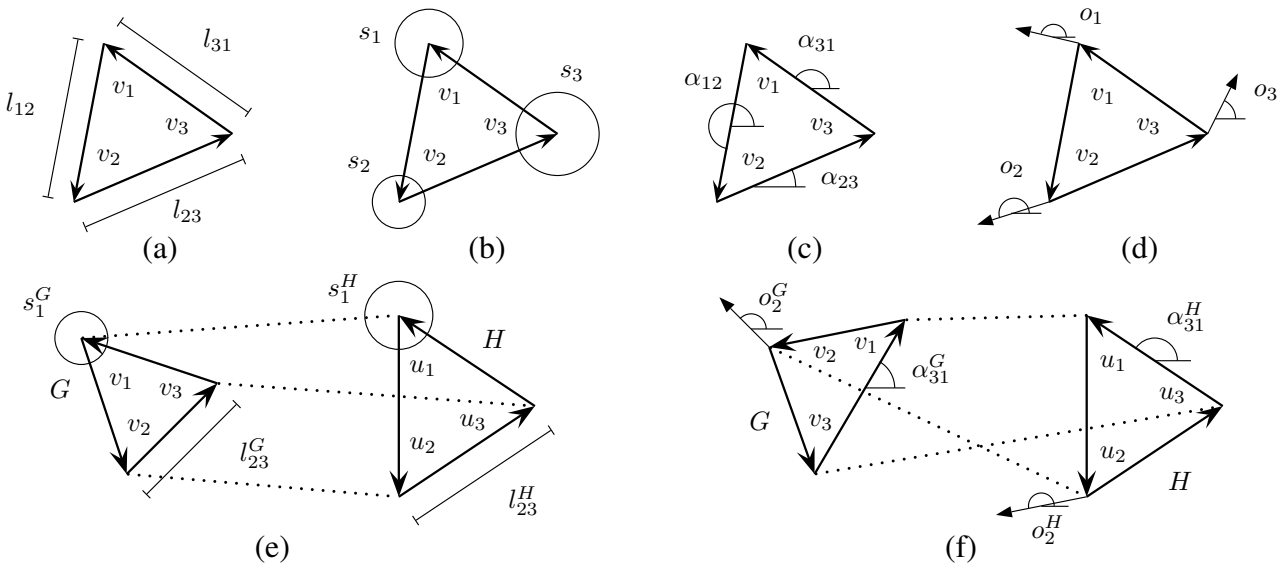


Figure 4. Structural information employed in keygraph match filtering. **(a)** Keygraph edges length: $l_{12}$, $l_{23}$ and $l_{31}$. **(b)** Keypoints (SIFT) scale: $s_1$, $s_2$ and $s_3$. **(c)** Keygraph edges orientation: $\alpha_{12}$, $\alpha_{23}$ and $\alpha_{31}$. **(d)** Keypoints (SIFT) orientation: $o_1$, $o_2$ and $o_3$. **(e)** Checking whether changes in keypoints scale and keygraph edges length are similar. For the corresponding keypoints scale $s_1^G$ (in a query keygraph $G$) and $s_1^H$ (in a training keygraph $H$), the ratio $\Phi_1 = s_1^G/s_1^H$ is calculated, as well as the ratio between the length of corresponding edges $\Phi_{23} = l_{23}^G/l_{23}^H$. We check whether $\Phi_1$ is similar to $\Phi_{23}$: $0.5 \leq \Phi_1/\Phi_{23} \leq 2$. A similar check is done for every pair $\Phi', \Phi''$ from the set of ratios $\{\Phi_1, \Phi_2, \Phi_3, \Phi_{12}, \Phi_{23}, \Phi_{31}\}$. **(f)** Checking whether changes in keypoints orientation and keygraph edges orientation are similar. For the corresponding keypoints orientation $o_2^G$ and $o_2^H$, the change in orientation $\Delta_2 = o_2^G - o_2^H$ is calculated, as well as the change in orientation of the corresponding edges $\Delta_{31} = \alpha_{31}^G - \alpha_{31}^H$. We check whether $\Delta_2$ is similar to $\Delta_{31}$: $\arccos(\Delta_2 - \Delta_{31}) \leq 60°$. A similar check is done for every pair $\Delta', \Delta''$ from the set of changes in orientation $\{\Delta_1, \Delta_2, \Delta_3, \Delta_{12}, \Delta_{23}, \Delta_{31}\}$.

as a regular vector in 2D space) [2]. The orientations of the edges of a keygraph $U = (h_1, h_2, h_3)$ are denoted as $\alpha_{12}^U$, $\alpha_{23}^U$ and $\alpha_{31}^U$ (see Figure 4-c).

- *Keypoints orientation*: Each keypoint in $U = (h_1, h_2, h_3)$ has an orientation assigned by SIFT. They are denoted as $o_1^U$, $o_2^U$ and $o_3^U$ (see Figure 4-d).

For each candidate keygraph match $(G, H)$, our method calculates the changes in orientations, lengths and scales from $G$ to $H$. A valid keygraph match must present a similar change in edges orientation and keypoints orientation, as well as a similar change in edges length and keypoints scale.

Let the candidate keygraph match $(G, H)$ between $G = (v_1, v_2, v_3)$ and $H = (u_1, u_2, u_3)$ be associated to the keypoint matches $\mathcal{M} = \{(v_1, u_1), (v_2, u_2), (v_3, u_3)\}$, as illustrated by Figure 4-e. Attribute changes are calculated as:

- *Changes in edges length*: Three ratios between the length of corresponding edges in the match $(G, H)$: $\Phi_{12} = l_{12}^G/l_{12}^H$, $\Phi_{23} = l_{23}^G/l_{23}^H$, $\Phi_{31} = l_{31}^G/l_{31}^H$;
- *Changes in keypoints scale*: Three ratios between the scale of corresponding keypoints in the match $(G, H)$: $\Phi_1 = s_1^G/s_1^H$, $\Phi_2 = s_2^G/s_2^H$, $\Phi_3 = s_3^G/s_3^H$;
- *Changes in edges orientation*: Three changes in orientation of corresponding edges in the match $(G, H)$: $\Delta_{12} = \alpha_{12}^G - \alpha_{12}^H$, $\Delta_{23} = \alpha_{23}^G - \alpha_{23}^H$, $\Delta_{31} = \alpha_{31}^G - \alpha_{31}^H$;
- *Changes in keypoints orientation*: Three changes in orientation of corresponding keypoints in $(G, H)$: $\Delta_1 = o_1^G - o_1^H$, $\Delta_2 = o_2^G - o_2^H$, $\Delta_3 = o_3^G - o_3^H$.

**Similar changes in keypoints scale and keygraph edges length**. When an image is subject to a zooming transformation with factor $r$, the length of every keygraph edge and keypoint scale is multiplied by $r$. But when a perspective transformation changing the viewing angle in $\psi$ degrees is employed, a unit circle becomes an ellipse whose longer and shorter axes have length 1 and $\cos\psi$, respectively [13].

SIFT features lose reliability when $\psi > 60°$ [8]. When $\psi = 60°$, the length of the transformed ellipse's shorter axe divided by the original circle's diameter is $\cos 60°/1 = 0.5$.

For a keygraph match $(G, H)$, let the three ratios between the length of corresponding edges be $\Phi_{12}$, $\Phi_{23}$ and $\Phi_{31}$, while the three ratios between the scale of corresponding keypoints are $\Phi_1$, $\Phi_2$ and $\Phi_3$. As illustrated in Figure 4-e, a valid keygraph match must satisfy the following property: for any pair $\Phi', \Phi''$ of those six ratios $\{\Phi_{12}, \Phi_{23}, \Phi_{31}, \Phi_1, \Phi_2, \Phi_3\}$, the largest ratio must be low-

[2] The origin of an image is in the bottom-left corner: the $y$ axis points upwards and the $x$ axis points to the right. Positive angles are counter-clockwise, starting from the $x$ axis.

ert than than twice the smaller ratio, *i.e.*

$$0.5 \leq \frac{\Phi'}{\Phi''} \leq 2. \quad (1)$$

**Similar changes in keypoints orientation and keygraph edges orientation**. Rotating an image by $\theta$ degrees changes the orientation of every keypoint and keygraph edge in this image in the same $\theta$ degrees. On the other hand, when a perspective transformation is applied, not every keypoint and keygraph edge rotates in the same $\theta$ degrees, although very different changes in orientation cannot occur.

For the keygraph match $(G, H)$, let the changes in keygraph edges orientation be $\Delta_{12}$, $\Delta_{23}$ and $\Delta_{31}$, while the changes in keypoints orientation are $\Delta_1$, $\Delta_2$ and $\Delta_3$. As illustrated in Figure 4-f, a valid keygraph match must satisfy the following property: for any pair $\Delta', \Delta''$ of those six changes in orientation $\{\Delta_{12}, \Delta_{23}, \Delta_{31}, \Delta_1, \Delta_2, \Delta_3\}$, the angle between $\Delta'$ and $\Delta''$ must be less than $60°$, *i.e.*,

$$\arccos(\Delta' - \Delta'') \leq 60°. \quad (2)$$

### 3.3. Third stage: pose estimation

In the third stage, the remaining keygraph matches are used to localise objects and estimate their pose.

One keygraph match generates $\kappa = 3$ keypoint matches, which can be used to instantiate a candidate affine transformation mapping query and training image. Each candidate pose is then evaluated by counting the number of keypoint (vertex) matches that agree with it.

**Affine transformations with agreeing keypoint matches**. Let $\mathcal{G}$ be the set of keygraph matches between the query image and a training image $I$, $\mathcal{G} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{|\mathcal{G}|}\}$, in which $\mathcal{M}_i$ is a set of three keypoint matches (associated to a keygraph match). Thus the set of *keypoint* matches between those images is $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \ldots \cup \mathcal{M}_{|\mathcal{G}|}$.

To evaluate the quality of a candidate affine transformation, the method counts the number of matches that agree with this transformation: for each keypoint match $(p, q) \in \mathcal{M}$, let $x_p, y_p$ be the position in the query image as established by the match, and let $f(x_q), f(y_q)$ be the position in the query image as predicted by the affine transformation. If the $\ell_\infty$ distance between $x_p, y_p$ and $f(x_q), f(y_q)$ is below *three* pixels, the keypoint match agrees with the candidate transformation. The confidence in a solution is given by such a number of keypoint match agreements.

Each affine transformation with at least one keypoint agreement is considered as a possible solution. Each possible solution is further refined using least-squares fit.

**Dealing with multiple detections**. The final stage involves projecting the objects and dealing with multiple detections.

First, the candidate affine transformations are sorted based on the number of agreeing keypoint matches (*i.e.* confidence). The most confident solution is returned as a correct pose, and the ground-truth of its matched training image is projected in the query image.

Then, the next more confident solution re-counts its agreeing keypoint matches, now discarding matches lying inside the projection of any previously returned solution. If at least one keypoint agreement remains, that solution is returned as a correct pose and the ground-truth of its training image is projected in the query image. The procedure continues by similarly examining the remaining elements in the sequence of produced possible solutions.

## 4. Experiments and results

For performance evaluation, we used the CMU10 object recognition dataset made available by Hsiao *et al.* [6]. This dataset contains ten types of training objects, for a total of 250 training images and 500 query images.

We also evaluated the scalability of our method, by significantly increasing the number of training images. We randomly selected a subset of 4000 images from the PASCAL VOC 2007 dataset [1] and employed them as training images together with the CMU10 training images. Detecting a PASCAL image constitutes an error.

SIFT keypoints were extracted using the VLFeat [14] library. The CMU10 and CMU10+PASCAL dataset generated $2.5 \times 10^5$ and $7.5 \times 10^6$ training keypoints, respectively.[3] For each dataset, the keypoints were indexed in a hierarchical $k$-means tree (both using $k = 16$).

For each detection, we used the recovered affine transformation to project the training object's ground truth segmentation onto the query image and calculated the region $R$ inside the convex hull. The region overlap criterion $(R \cap R_{\text{gt}})/(R \cup R_{\text{gt}}) > 0.4$ between the region $R$ and the ground truth segmentation $R_{\text{gt}}$ (in the query image) determines if an object is correctly detected.[4] To evaluate the results we used the mean Average Precision (AP), computed over all the ten object classes of this dataset.

### 4.1. Baseline systems

We evaluated our method against SCRAMSAC [13], Lowe's test [8] and the results reported by Hsiao *et al.* [6].

[3] For the CMU10 images, we used ground-truth segmentation to remove keypoints not belonging to the object. For the PASCAL images, ground-truth was not used, thus all parts of an image act as a training "object".

[4] Hsiao *et al.* [6] projected 3D models onto 2D query images; their 3D models included inferred object faces which are not actually visible in any training image. In order to make a more fair evaluation of our method, which is based on affine transformations, we modified the ground-truth segmentations in the query images by removing hidden faces of the object (e.g. the top of tins). Furthermore, we observed that our method enables the use of a smaller overlap threshold (0.4, instead of 0.5), giving more true positive matches without any increase in the number of false positives.

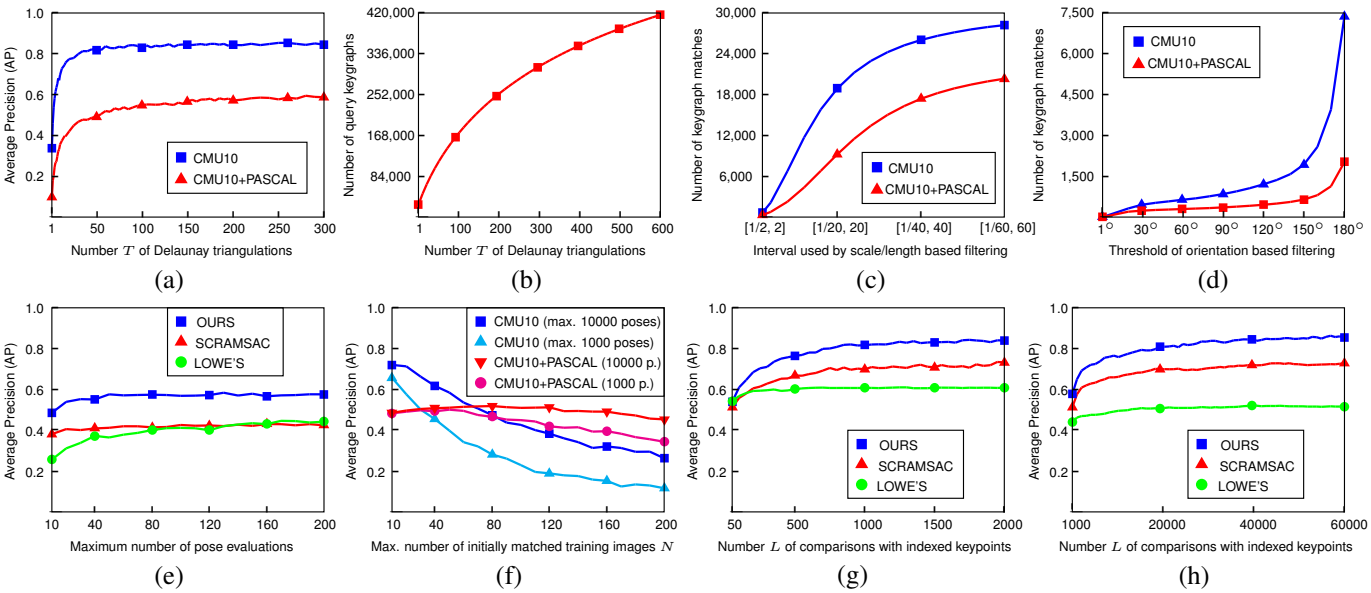


Figure 5. **First row**: evaluation of the effect of parameters in our method. **(a)** AP versus number $T$ of Delaunay triangulations for the CMU10 or CMU10+PASCAL training datasets. In the latter dataset, PASCAL images were added to the training set to assess scalability and the effect of added negative samples, which is expected to reduce the AP as more mismatches may occur. **(b)** Average number of keygraphs sampled in a query image versus $T$. **(c)** and **(d)** Average number of keygraph matches between a query image and the training images versus the threshold used by the filtering stage based on (c) scale/length or (d) orientation. **Second row**: comparisons against other methods. **(e)** AP versus the maximum allowed number of pose evaluations of all training images for each query. **(f)** AP of SCRAMSAC versus the maximum allowed number $N$ of training images matched by a query keypoint. **(g)** and **(h)** AP versus the number $L$ of keypoint comparisons between query and indexed training keypoints, for (g) CMU10 or (h) CMU10+PASCAL.

Initially, a hierarchical $k$-means tree is used to establish keypoint matches. For Lowe's test, each query keypoint $p$ can only match its single nearest neighbor in the whole training set. For SCRAMSAC, each query keypoint $p$, after running through the tree, can match keypoints in different training images, up to one match per image. Then, for each query keypoint $p$, only the top $N$ keypoint matches are retained (based on descriptor distance). Finally, keypoint matches without sufficient local support are removed [13].

In the final stage, RANSAC is used for object pose estimation. For Lowe's test and SCRAMSAC, each putative affine transformation is generated by randomly selecting *three* keypoint matches that occur with the same training image. For our method, each putative affine transformation is generated by randomly selecting *one* keygraph match.

## 4.2. Results

Figure 6 shows two sample object detection results of our method.

Figure 5-a shows the AP achieved by our method versus the number $T$ of Delaunay triangulations used to sample keypoint triples. Each curve corresponds to a training dataset, CMU10 or CMU10+PASCAL; in both cases, in the initial keypoint matching stage, $L = 1000$ training keypoints were compared to each query keypoint.

As shown in Figure 5-a, employing a large number of Delaunay triangulations provided a significant gain in performance: the AP increased in almost $0.50$ in comparison to using $T = 1$. For the CMU10 dataset, the AP plateaued at $T = 50$. Interestingly, when employing the more complex CMU10+PASCAL training dataset, the AP continued to increase for $T > 50$, going from $0.49$ to $0.58$ when using $T = 200$. This suggests that our sampling strategy was able to generate a diverse set of query keygraphs, as employing more triangulations partially compensated the decrease in accuracy due to the use of a larger number of training images.

Figure 5-b shows that the number of keygraphs grows sub-linearly with the number of triangulations.

Figure 5-c shows the average number of keygraph matches between a query image and all the training images versus the threshold of the filtering stage in Equation 1. We changed the interval of accepted ratios, which is $[0.5, 2]$ in Equation 1. When a large interval of $[1/60, 60]$ is used, all candidates are accepted, thus filtering is performed only by the cyclic ordering requirement and the orientation based filtering stage (which uses a threshold of $60°$). Increasing the interval from $[0.5, 2]$ to $[1/60, 60]$ caused the number of established keygraph matches to increase by a factor of $100$.

Figure 5-d is similar to Figure 5-c, but considers the orientation based filtering stage. Increasing the threshold from $60°$ (Equation 2) to $180°$ caused the number of established

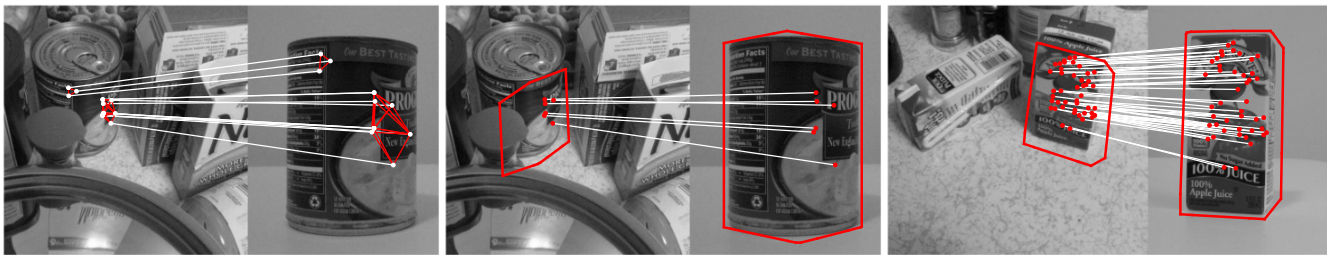

Figure 6. Keygraph and keypoint matches. The region overlaps achieved by the affine transformations were 0.61 and 0.73, respectively.

keygraph matches to increase by a factor of 10.

Using both Equations 1 and 2 for filtering, the average number of keygraph matches established per query image was 600, for the CMU10 training images ($T = 50$), and 400, for the CMU10+PASCAL training images ($T = 200$).

Figure 5-e shows the AP versus the maximum allowed number of pose evaluations (affine transformations), employing the CMU10+PASCAL training dataset. Our method achieved a better performance than SCRAMSAC and Lowe's test, even for very few pose evaluations.

Figure 5-f shows the AP achived by SCRAMSAC versus the maximum allowed number of matches $N$ between a query keypoint and the training images. We evaluated two cases: using a maximum of 1000 or 10000 pose evaluations. Using the CMU10 training dataset, the best results were achieved around $N = 10$, while for the more complex CMU10+PASCAL training dataset, the best results were achieved around $N = 50$; in both cases, using additional pose evaluations improved the performance. An advantage of the proposed keygraphs method is not having a parameter such as $N$, as our method does not limit the number of training images initially matched by a query keypoint.

Figure 5-g shows the AP versus the number $L$ of indexed training keypoints which are compared to a query keypoint, for the CMU10 training dataset. Good results were achieved by employing $L = 1000$ (0.4% of the number of stored keypoints), for both SCRAMSAC and our method. For Lowe's test, setting $L$ above 400 did not provide significant improvement.

Figure 5-h is similar to Figure 5-g, but considers the CMU10+PASCAL dataset. For our method and SCRAMSAC, employing $L = 30000$ allows achieving the same AP as when using $L = 1000$ with the CMU10 dataset. This is interesting, as the number of training descriptors, from CMU10 to CMU10+PASCAL, also increased by a factor of 30. On the other hand, Lowe's test was not able to similarly compensate the performance loss caused by the use of a larger number of training images.

In summary, for the CMU10 training dataset, the AP of our method, SCRAMSAC and Lowe's test was, respectively, 0.82, 0.70 and 0.61. For the CMU10+PASCAL training dataset the AP was: for $L = 1000$: 0.58, 0.51 and 0.44; for $L = 2000$: 0.65, 0.58 and 0.46; and for $L = 30000$: 0.84, 0.70 and 0.51.

The AP achieved by our method using the CMU10 training dataset, 0.82, was superior to the AP reported by Hsiao *et al*. [6], 0.78, even though Hsiao *et al*. employed 3D object models which allows to use matches to different training images and to use 3D-2D pose instantiation.

We evaluated the computation time demanded by our method (using a C implementation). For both the CMU10 and CMU10+PASCAL training dataset, when using $L = 1000$, $T = 50$, the time per query image was $1.4 \pm 0.3$ sec., with 55% of the time being spent on keypoint matching, 3% on keygraph sampling and 39% on keygraph matching. Using the CMU10+PASCAL training images and $T = 200$, the time per query image was, for $L = 1000$, $2.5 \pm 0.7$ sec., and for $L = 2000$, $6.8 \pm 1.7$ sec.; for $L = 1000$, 33% of the time was spent on keypoint matching, 9% on keygraph sampling and 57% on keygraph matching; for $L = 2000$ those values were 26%, 3% and 70%, respectively.

## 5. Conclusion

Keygraphs constitute an intermediate level feature, above the keypoints. Initially, keypoint matches are established by using an indexing tree. Then, keypoint matches are transformed into keygraph matches, which correctly removes the large majority of the initial keypoint matches.

This paper introduced structural properties used to filter out candidate keygraph matches. We considered an heterogeneous information set, concerning keypoints scale, orientation, image position and cyclic ordering, which allows to correctly filter out 99% of the candidate keygraph matches.

We also introduced an efficient algorithm to sample query keypoint triples, based on using complementary Delaunay triangulations. Each triangulation is done in a way that vertices are not too close to each other so the triangles generate more robust pose hypotheses.

At application, the processing time of our method depends almost exclusively on parameters that are pre-set by the user (triangulations $T$ and keypoint comparisons $L$). We showed that a 30-fold increase in the number of training keypoints does not significantly deteriorate the performance of our method.

## References

[1] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 6

[2] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2

[3] Q. Hao, R. Cai, Z. Li, L. Zhang, Y. Pang, and F. Wu. 3d visual phrases for landmark recognition. In *Proc of the IEEE Conf on Computer Vision and Pattern Recognition, CVPR*, pages 3594–3601, 2012. 1, 2

[4] Q. Hao, R. Cai, Z. Li, L. Zhang, Y. Pang, F. Wu, and Y. Rui. Efficient 2d-to-3d correspondence filtering for scalable 3d object recognition. In *Proc of the IEEE Conf on Computer Vision and Pattern Recognition, CVPR*, pages 899–906, 2013. 1, 2

[5] M. Hashimoto and R. M. Cesar Jr. Object detection by keygraph classification. In *Graph-Based Representations in Pattern Recognition*, pages 223–232. Springer, 2009. 2

[6] E. Hsiao, A. Collet, and M. Hebert. Making specific features less discriminative to improve point-based 3d object recognition. In *Proc of the IEEE Conf on Computer Vision and Pattern Recognition, CVPR*, pages 2653–2660, 2010. 1, 2, 6, 8

[7] H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87(3):316–336, 2010. 2

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004. 1, 2, 3, 5, 6

[9] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005. 1, 2

[10] H. Morimitsu, R. B. Pimentel, M. Hashimoto, R. M. Cesar, and R. Hirata. Wi-fi and keygraphs for localization with cell phones. In *Proc of ICCV Workshops*, pages 92–99. IEEE, 2011. 2, 3

[11] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 331–340, 2009. 2, 3

[12] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc of the IEEE Conf on Computer Vision and Pattern Recognition, CVPR*, pages 1–8, 2007. 1, 2

[13] T. Sattler, B. Leibe, and L. Kobbelt. SCRAMSAC: Improving RANSAC's efficiency with a spatial consistency filter. In *Proc Int Conf on Computer Vision, ICCV*, pages 2090–2097, 2009. 1, 2, 5, 6, 7

[14] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008. 6

# Referências Bibliográficas


**Everingham** *et al.* **(2010)** Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn e Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338. Citado na pág. 31

**Fischler e Bolles (1981)** Martin A Fischler e Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395. Citado na pág. 10

**Gordon e Lowe (2006)** Iryna Gordon e David G Lowe. What and where: 3d object recognition with accurate pose. Em *Toward category-level object recognition*, páginas 67–82. Springer. Citado na pág. 10, 14, 16

**Hao** *et al.* **(2012)** Qiang Hao, Rui Cai, Zhiwei Li, Lei Zhang, Yanwei Pang e Feng Wu. 3d visual phrases for landmark recognition. páginas 3594–3601. Citado na pág. 15, 16, 17

**Hao** *et al.* **(2013)** Qiang Hao, Rui Cai, Zhiwei Li, Lei Zhang, Yanwei Pang, Feng Wu e Yong Rui. Efficient 2d-to-3d correspondence filtering for scalable 3d object recognition. páginas 899–906. Citado na pág. 14, 15, 17, 18

**Hashimoto e Cesar Jr (2009)** Marcelo Hashimoto e Roberto M Cesar Jr. Object detection by keygraph classification. Em *Graph-Based Representations in Pattern Recognition*, páginas 223–232. Springer. Citado na pág. 2, 17

**Hsiao** *et al.* **(2010)** Edward Hsiao, Alvaro Collet e Martial Hebert. Making specific features less discriminative to improve point-based 3d object recognition. páginas 2653–2660. Citado na pág. vi, vii, 10, 11, 16, 31, 32, 33, 44

**Jégou** *et al.* **(2010)** Hervé Jégou, Matthijs Douze e Cordelia Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87(3):316–336. Citado na pág. 12, 13, 14

**Lowe (2001)** David G Lowe. Local feature view clustering for 3d object recognition. volume 1, páginas I–682. IEEE. Citado na pág. 36

**Lowe (2004)** David G Lowe. Distinctive image features from scale-invariant keypoints. 60(2): 91–110. Citado na pág. 9, 10, 12, 28, 34, 40, 42, 44

**Mikolajczyk e Schmid (2005)** Krystian Mikolajczyk e Cordelia Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27 (10):1615–1630. Citado na pág. 11

**Moravec (1981)** Hans P Moravec. Rover visual obstacle avoidance. Em *International Joint Conference on Artificial Intelligence*, páginas 785–790. Citado na pág. 6

**Morimitsu** *et al.* **(2011)** Henrique Morimitsu, Rodrigo B Pimentel, Marcelo Hashimoto, Roberto M Cesar e R Hirata. Wi-fi and keygraphs for localization with cell phones. Em *Proc of ICCV Workshops*, páginas 92–99. IEEE. doi: 10.1109/ICCVW.2011.6130228. Citado na pág. 15, 17, 18, 22

**Muja e Lowe (2009)** Marius Muja e David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. Em *International Conference on Computer Vision Theory and Applications (VISAPP)*, páginas 331–340. Citado na pág. 19, 21

**Philbin** *et al.* **(2007)** James Philbin, Ondrej Chum, Michael Isard, Josef Sivic e Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. páginas 1–8. Citado na pág. 11

**Sattler** *et al.* **(2009)** Torsten Sattler, Bastian Leibe e Leif Kobbelt. SCRAMSAC: Improving RANSAC's efficiency with a spatial consistency filter. páginas 2090–2097. Citado na pág. 12, 28, 34

**Schmid e Mohr (1997)** Cordelia Schmid e Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–534. Citado na pág. 6, 7

**Simonyan** *et al.* **(2014)** Karen Simonyan, Andrea Vedaldi e Andrew Zisserman. Learning local feature descriptors using convex optimisation. 2(4). Citado na pág. 48

**Vedaldi e Fulkerson (2008)** A. Vedaldi e B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008. Citado na pág. 32

**Winder** *et al.* **(2009)** Simon Winder, Gang Hua e Matthew Brown. Picking the best daisy. Em *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, páginas 178–185. IEEE. Citado na pág. 18

**Zhang** *et al.* **(1995)** Zhengyou Zhang, Rachid Deriche, Olivier Faugeras e Quang-Tuan Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial intelligence*, 78(1):87–119. Citado na pág. 6

**Zitnick** *et al.* **(2007)** C Lawrence Zitnick, Jie Sun, Richard Szeliski e Simon Winder. Object instance recognition using triplets of feature symbols. Relatório técnico, Technical report, Microsoft Research, Redmond, WA, USA. Citado na pág. 15