UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

3D Facial Reconstruction from Point Clouds with Hermite Radial Basis Functions and Multilevel Partition of Unity

Guilherme Hideo Tubone

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Guilherme Hideo Tubone

3D Facial Reconstruction from Point Clouds with Hermite Radial Basis Functions and Multilevel Partition of Unity

Dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. João do Espírito Santo Batista Neto

USP – São Carlos
January 2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

T8853

Tubone, Guilherme
3D Facial Reconstruction from Point Clouds with Hermite Radial Basis Functions and Multilevel Partition of Unity / Guilherme Tubone; orientador João do Espírito Santo Batista Neto. -- São Carlos, 2024.
64 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. Point clouds. 2. Surface reconstruction. 3. Faces. 4. Radial Basis Functions. 5. Multilevel Partition of Unity. I. do Espírito Santo Batista Neto, João, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Guilherme Hideo Tubone

Reconstrução Facial 3D a Partir de Nuvem de Pontos com Funções de Base Radial de Hermite e Partição da Unidade Multinível

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *EXEMPLAR DE DEFESA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. João do Espírito Santo Batista Neto

USP – São Carlos
Janeiro de 2024

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my family for the support and encouragement they have given me throughout my life. I would like to express my appreciation to my girlfriend, Luiza, for her love and for always being by my side. I also would like to express my gratitude to my friends for their support. Finally, I would like to thank my advisor, João Batista, for providing me with the opportunity and guidance to develop this work.

ABSTRACT

TUBONE, G. H. **3D Facial Reconstruction from Point Clouds with Hermite Radial Basis Functions and Multilevel Partition of Unity**. 2024. 64 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

Point cloud surface reconstruction is the process of transforming raw data from 3D scanners into a digital representation of an object surface. This process has applications in various fields, and as a result, many methods have been proposed to solve this problem over the years. In this work, we focus on methods that generate surfaces represented as level sets of implicit functions. Specifically, we propose a method based on Hermite Radial Basis Functions (HRBF) with Multilevel Partition of Unity (MPU). The HRBF allows a precise and detailed representation of the surface geometry, even when dealing with nonuniform data, while the MPU enables a efficient and adaptive reconstruction process. To demonstrate the efficacy of our method, we applied it to a dataset of human faces. The results show that the method can successfully reconstruct the facial geometry, capturing details of the surfaces.

Keywords: Point clouds, Surface reconstruction, Faces, Hermite Radial Basis Functions, Multilevel Partition of Unity.

RESUMO

TUBONE, G. H. **Reconstrução Facial 3D a Partir de Nuvem de Pontos com Funções de Base Radial de Hermite e Partição da Unidade Multinível**. 2024. 64 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

A reconstrução de superfície de nuvem de pontos é o processo de transformar dados brutos de scanners 3D em uma representação digital da superfície de um objeto. Esse processo tem aplicações em vários campos, e por isso, muitos métodos têm sido propostos ao longo dos anos para resolver esse problema. Neste trabalho, nos concentramos em métodos que geram superfícies representadas como níveis de funções implícitas. Especificamente, propomos um método baseado em Funções de Base Radial de Hermite (HRBF) com Partição de Unidade Multinível (MPU). A HRBF permite uma representação precisa e detalhada da geometria da superfície, mesmo quando se trabalha com dados não uniformes, enquanto o MPU permite um processo de reconstrução eficiente e adaptativo. Para demonstrar a eficácia do nosso método, aplicamos-no a um conjunto de dados de faces humanas. Os resultados mostram que o método pode reconstruir com sucesso a geometria facial, capturando detalhes das superfícies.

Palavras-chave: Nuvem de pontos, Reconstrução de superfícies, Faces, Funções de Base Radial de Hermite, Partição da Unidade Multinível.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

CHAPTER

1

INTRODUCTION

Recent advances in 3D scanners have enabled the collection of high-resolution information about object surfaces. However, the raw point clouds generated by these scanners need to be converted into digital representations of the scanned surfaces, a process known as surface reconstruction. Surface reconstruction finds applications in various fields, such as computer-aided design, computer graphics, computer vision, and medical imaging (LU *et al.*, 2005). Given the relevance of this task, many methods have been proposed over the years (BERGER *et al.*, 2017).

There are two main types of surface representations: explicit and implicit. Explicit representations can be a triangulation of the initial set of points or approximated by a parametric equation, while implicit representations are defined by a function such that one of its level sets is the surface approximation. Implicit representations require post-processing, such as the Marching Cubes algorithm (LORENSEN; CLINE, 1987), to be visualized.

One implicit approach to surface reconstruction is the Radial Basis Function (RBF) Interpolation. There are many works describing the usage of Radial Basis Functions (RBF) for surface reconstruction, such as Carr, Fright and Beatson (1997), Fasshauer (2007), Morse *et al.* (2005), Turk and O'brien (2002). RBF interpolation has been widely used for implicit surface reconstruction, but it can be time-consuming for large point clouds due to the need to solve a linear system of equations. This issue has been addressed by several techniques, including center reduction, the Fast Multipole Method (FMM) (CARR *et al.*, 2001), and the usage of Compactly Supported Radial Basis Functions (WU, 1995; WENDLAND, 1995).

Later, the Multilevel Partition of Unity (MPU) method was presented by Ohtake *et al.* (2003) as another implicit solution that is able to deal with large point sets. The core idea of this method is to partition the point set domain in several sub-domains, then solve the local problems by finding local interpolants and, finally, mix these local solutions into a global interpolant.

The Multilevel Partition of Unity (MPU) method, introduced by (OHTAKE *et al.*, 2003), is another implicit solution for handling large point sets in surface reconstruction. The MPU

method partitions the point set domain into several sub-domains, solves local problems by finding local interpolants, and combines these local solutions into a global interpolant. This approach can also reduce the computational cost and memory requirements of RBF interpolation, making it more practical for large point clouds.

Another issue with regular RBF interpolation is the need to create extra points on the normal vectors direction to the initial point set before performing interpolation. This problem is addressed by Hermite Radial Basis Function Implicits (HRBF), which was introduced by Macedo, Gois and Velho (2011). HRBF offers better performance than previous methods in handling nonuniform samples and close sheets.

In this work, we propose a method that combines the advantages of Hermite Radial Basis Function Implicits (HRBF) and the efficiency provided by the Multilevel Partition of Unity (MPU). The 2D/3D Florence Face Dataset (BAGDANOV; BIMBO; MASI, 2011) was used to evaluate the performance of the method, and it was found to accurately reconstruct facial geometry with details in a reasonable time and memory usage.

1.1 Contributions

The main contribution of this MSc project is to perform surface reconstruction from point clouds using HRBF Interpolation along with Multilevel Partition of Unity. This approach makes it possible to reconstruct surfaces with all benefits of HRBF, but with less time and memory consumption. This idea was suggested by Macedo, Gois and Velho (2011), but until now, it hasn't been implemented.

Also, being part of a FAPESP project called "Mapeamento de características robustas entre diferentes domínios e espaços R^2 e R^3", this project performed surface reconstruction for a dataset of human faces, the 2D/3D Face Dataset from The Media Integration and Communication Center (MICC) and University of Florence.

1.2 Content of the manuscript

This work is organized as follows:

Chapter 2 Discusses related work in the field of surface reconstruction, more specifically: Global Supported RBF Interpolation, Compactly Supported RBF Interpolation, Hermite RBF Interpolation and Partition of Unity.

Chapter 3 Introduces the necessary theoretical background to understand the work: Radial Basis Function Interpolation, Hermite Radial Basis Functions Interpolation, 2^n-trees (Quadtree, Octree), Multilevel Partition of Unity and Marching Cubes algorithm.

Chapter 4 Describes how the theory described in the last chapter was used to solve the surface reconstruction problem.

Chapter 5 Presents the results of our work.

Chapter 6 Summarizes the main contributions and findings of our work, and discusses the implications of our results for the field of surface reconstruction. It also provides suggestions for future research in this area.

CHAPTER

2

RELATED WORK

In this chapter we will explore relevant research in our area of interest. First, we will delve into surface reconstruction from point clouds with Global Supported Radial Basis Functions (GSRBF) and Compactly Supported Radial Basis Function (CSRBF) interpolations, then we will present pieces of reaserch that employ Hermite Radial Basis Function (HRBF) interpolation and Multilevel Partition of Unity (MPU).

An extensive review for implicit surface reconstruction via RBF was recently conducted by Mo, Shou and Chen (2022). Some of the discussion presented in this chapter is based on their work.

2.1 Global Supported RBF interpolation

Hoppe *et al.* (1992) described and demonstrated an algorithm to solve the problem of taking an unorganized set of points $\{x_1, ..., x_n\} \subset \mathbb{R}^3$ from a surface M, yielding an approximation M' of it as a simplicial surface. The solution was based on finding the zero set of a signed distance function. Later, Savchenko *et al.* (1995) suggested a new approach to reconstruct surfaces from point clouds, by interpolating it via a linear system with RBF.

Carr, Fright and Beatson (1997) proposed a practical solution to a problem of medical imaging using RBF. The problem was to interpolate incomplete surfaces from cranial implants in order to repair defects, such as holes, in the skull. A similar approach was used by Turk and O'brien (1999) to create smooth implicit surfaces of arbitrary topology.

Both works mentioned in the previous paragraph provided the following way to represent the interpolant: $f(x) = p(x) + \sum_{i=1}^{N} \alpha_i \phi(|x - x_i|)$, where $p(x)$ is a polynomial of degree k, α_i are coefficients, x_i are the centers of the RBF and $||$ is the euclidian norm in $\mathbb{R}^3$. From this formulation, it is possible to find the coefficients by solving a linear system of equations.

These works that make use of Global Supported Radial Basis Functions (GSRBF) were

later improved by several authors in combination with more efficient methods, such as the ones proposed by Carr *et al.* (2001) and Tobor, Reuter and Schlick (2004). The prior uses FMM (Fast Multipole Method) (GREENGARD; ROKHLIN, 1987) to fastly evaluate RBF and Center Reduction - a technique to reduce the initial set of points aiming at faster interpolation, while keeping accuracy. The latter employs a multi-scale approach, where the domain of the given point cloud is subdivided by means of a balanced binary tree (solved separately) and then blended together with partition of unity.

2.2 Compactly Supported RBF interpolation

One of the problems related to GSRBF is that the interpolation matrix is completely filled and when applied to a big set of points exhibits a high computational cost. Moreover, small changes in the values can propagate as large changes in the final result. In the light of such limitations, Wendland (1995) and Buhmann (1998) proposed CSRBF (Compactly Supported RBF), functions with finite support, which means they are nonzero only for a certain distance from the center. As a result, CSRBF will have a better local interpolation, with lower computational cost.

CSRBF has been widely used in implicit surface reconstruction for very large point clouds. Morse *et al.* (2001), for example, introduced CSRBF for surface reconstruction and showed that it could be used along with computational methods and data structures for spatial subdivision. Ohtake, Belyaev and Seidel (2003) presented a coarse-to-fine hierarchical method for surface reconstruction and Liu *et al.* (2018) introduced a method to generate implicit surfaces from polygon soups.

2.3 Hermite RBF intepolation

The reconstruction of implicit surfaces from scattered points using RBF interpolation often requires offset-points in the direction of the normals in order to avoid the trivial solution. However, this is not ideal as the tuning of parameters is sometimes necessary to properly find the ideal offset-points for the interpolation.

Macedo, Gois and Velho (2011) proposed the HRBF Implicits, a method to reconstruct an implicit functions that interpolates Hermite data (i.e., scattered points and their normal vectors). The HRBF Implicits is a particular case of the Hermite-Birkoff interpolation (WENDLAND, 2004) with RBF and it is proven to be fairly efficient when compared to similar approaches. It is particularly suitable when the set of point are irregular or coarse or also in the presence of close sheets.

The original HRBF Implicits have been further improved and used in different contexts. Brazil *et al.* (2011) employed them to render implicit surfaces in various pen-and-ink styles.

Gois *et al.* (2013) proposed the reconstruction of implicit surfaces from polygonal meshes, while Batagelo and Gois (2013) came up with a faster implementation by selecting a small subset of point-normals in a adaptive way. Finally, Trevisan, Gois and Batagelo (2014) used the Conjugate Gradients Method on GPU to solve the linear systems.

Nonetheless, the work done by Romeiro *et al.* (2014) is the one which best resembles the approach implemented in our project. The aforementioned used HRBF in forensic facial reconstruction, where faces of individuals are rebuilt from their skulls. The method uses anatomical rules to reconstruct the face surface by HRBF deformation over a mesh.

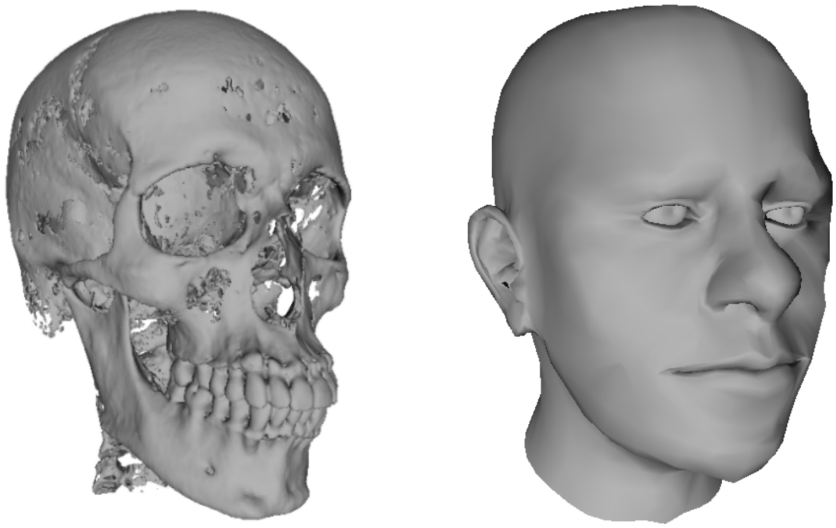


Figure 1 – Input skull and output facial reconstruction. (ROMEIRO *et al.*, 2014)

2.4 Partition of Unity

In order to reconstruct surface models from very large datasets, Ohtake *et al.* (2003) proposed a new shape representation called the multi-level partition of unity. They used piecewise quadratic functions to approximate the local shape of the surfaces, weight functions to blend these local approximations and an octree subdivision algorithm that adapts to the complexity of the local shape. Later Ohtake, Belyaev and Seidel (2006) improved the method by combining least-squares RBF and PU, achieving high quality models even in the presence of noisy and imcomplete data.

Then, Gois *et al.* (2008) introduced the Twofold Adaptive Partition of Unity Implicits, a method that combines the Partition of Unity Implicits with a space decomposition by tetrahedra based in the J_1^a triangulation (CASTELO *et al.*, 2006). And more recently, Drake, Fuselier and Wright (2022) presented a method similar to the approach used in our work. Their work is grounded on curl-free RBFs based on polyharmonic splines and they use partition of unity to make the method efficient and able to adapt to local shape features.

CHAPTER

3

THEORETICAL BACKGROUND

In this chapter, we present the fundamental concepts that serve as the foundation for the creation of our proposed project, which focuses on the reconstruction of surfaces from point clouds. We begin by providing essential mathematical definitions, followed by an explanation of Radial Basis Function interpolation and Hermite Radial Basis Function interpolation. We then introduce the 2^n-tree data structure, with a special focus on Quadtree and Octree and their application in space partitioning. Next, we discuss Multilevel Partition of Unity with Octree and conclude with the Marching Cubes algorithm, which is used to extract meshes from isosurfaces.

3.1 Basic concepts

In this section, we introduce the mathematical definitions and terminologies that are crucial to our project. To further explore some of the concepts presented in this section, we recommend the following books: Lima (2011), Bachman and Narici (2000).

Definition 3.1.1 (Euclidian Space)**.** A Euclidian Space of dimension n is the space $\mathbb{R}^n$ equipped with the usual Euclidian metric, that is, given two points $p = (p_1, ..., p_n)$ and $q = (q_1, ..., q_n) \in \mathbb{R}^n$, we measure their distance by:

$$||p - q|| = \sqrt{(p_1 - q_1)^2 + ... + (p_n - q_n)^2}$$

Definition 3.1.2 (Homeomorphism)**.** Let $U \subset \mathbb{R}^n$ and $V \subset \mathbb{R}^m$. A function $f\colon U \to V$ is a homeomorphism if it is a bijection that is continuous and has a continuous inverse. If such a function f exists, U and V are homeomorphic.

Definition 3.1.3 (Diffeomorphism)**.** A function $f\colon U \to V$ is a diffeomorphism of class C^r if it is a bijective, differentiable function with a differentiable inverse, and both f and f^{-1} have continuous derivatives up to order r. If such a function f exists, U and V are diffeomorphic.

Definition 3.1.4 (Manifold)**.** A set $M \subset \mathbb{R}^k$ is an n-dimensional manifold of class C^r if for every point $x \in M$, there is an open neighborhood $U \ni x$, with $U \subset \mathbb{R}^k$, an open subset V of $\mathbb{R}^{n-1} \times \mathbb{R}^+$, and a diffeomorphism of class C^r, $\phi : U \cap M \to V$.

Definition 3.1.5 (Surface)**.** A surface $S \subset \mathbb{R}^3$ is a 2-dimensional manifold in $\mathbb{R}^3$.

Definition 3.1.6 (Isosurface)**.** An Isosurface is a surface defined by the implicit function $f(x,y,z) = c$. In other words, it is a level set of a continuous function in $\mathbb{R}^3$.

Definition 3.1.7 (Radial Basis Function)**.** A Radial Basis Function is a real-valued function $\phi(x) : \mathbb{R}^d \to \mathbb{R}$, such that $\phi(x) = \hat{\phi}(||x-c||)$. In other words, its value depends on the distance between x and another point c, called a center.

Definition 3.1.8 (Inner product)**.** Let V be a vector space over the field of scalars $\mathbb{F}$. An *Inner Product* on V is a function $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{F}$ that satisfies the following properties for all vectors $\mathbf{x}, \mathbf{y}$, and $\mathbf{z} \in V$ and scalars $a, b \in \mathbb{F}$:

1. Linearity in the first argument: $\langle a\mathbf{x} + b\mathbf{y}, \mathbf{z} \rangle = a\langle \mathbf{x}, \mathbf{z} \rangle + b\langle \mathbf{y}, \mathbf{z} \rangle$.
2. Conjugate Symmetry: $\langle \mathbf{x}, \mathbf{y} \rangle = \overline{\langle \mathbf{y}, \mathbf{x} \rangle}$, where $\overline{\langle \mathbf{y}, \mathbf{x} \rangle}$ denotes the complex conjugate of $\langle \mathbf{y}, \mathbf{x} \rangle$.
3. Positive-definiteness: $\langle \mathbf{x}, \mathbf{x} \rangle > 0$, if $\mathbf{x} \neq \mathbf{0}$.

Definition 3.1.9 (Linear Functional)**.** A linear functional is a linear map from the vector space to its corresponding field of scalars.

Definition 3.1.10 (Hilbert Space)**.** A Hilbert Space $\mathbb{H}$ is a vector space that has an inner product and is complete in regards to the metric derived from its inner product.

Definition 3.1.11 (Dual Space)**.** The Dual Space $\mathbb{H}^*$ of a real Hilbert Space $\mathbb{H}$ is the set of all continuous linear functionals from $\mathbb{H}$ to its field of scalars.

Definition 3.1.12 (Generalized interpolant)**.** Consider a set of linearly independent functionals $\Lambda = \{\lambda_1, ..., \lambda_n\} \subseteq \mathbb{H}^*$ and a set of values $\{f_i \in \mathbb{R} : i = 1, ..., n\}$. A generalized interpolant $s \in \mathbb{H}$ is a function such that $\lambda_i(s) = f_i, 1 \leq i \leq n$.

Definition 3.1.13 (Gradient of a function)**.** The gradient of a function $f : \mathbb{R}^3 \to \mathbb{R}$, denoted as ∇f, is given by: $\nabla f(x,y,z) = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})$.

Definition 3.1.14 (Support)**.** The support of a function f is the closure of the set $\{x : f(x) \neq 0\}$, and is denoted by $supp(f)$.

Definition 3.1.15 (Compact set)**.** In the Euclidian Space, a compact set is defined as a closed and bounded set.

Definition 3.1.16 (Compact support)**.** A function f has a compact support if $supp(f)$ is a compact set.

Definition 3.1.17 (Graph)**.** A graph G is defined by (V,E), where V is a set of vertices and E is a set of edges, connecting those vertices.

Definition 3.1.18 (Tree)**.** A tree is a acyclic and connected graph.

3.2 Surface reconstruction with RBF

3.2.1 Problem formulation

Task: Given a point cloud, that is, n distinct points $\{\mathbf{x_i}\}_{i=1}^{n}$ of a surface M, find a surface M', that is an approximation of M. In this case, an implicit function $f(\mathbf{x})$, such that $M' = f^{-1}(0)$.

That means this method defines a surface implicitly by a function $f : \mathbb{R}^3 \to \mathbb{R}$, where each of the surface points satisfies $f(\mathbf{x_i}) = 0$.

However, with these conditions alone a function like $s(\mathbf{x}) = 0$ would be a trivial solution. Therefore, it is necessary to add some extra points, like shown in Figure 2, that do not belong to the surface. Hence:

- $f(\mathbf{x}) = 0$, on the surface
- $f(\mathbf{x}) < 0$, inside the surface
- $f(\mathbf{x}) > 0$, outside the surface

For each point of the initial set, two more points will be added, both in the direction of the normal vector of that point, one inside and another outside the surface:

$$\mathbf{x_i}^{in} = \mathbf{x_i} - \varepsilon\vec{n} \tag{3.1}$$

$$\mathbf{x_i}^{out} = \mathbf{x_i} + \varepsilon\vec{n} \tag{3.2}$$

And given $\delta_i > 0$:

$$f(\mathbf{x_i}^{in}) = -\delta \tag{3.3}$$

$$f(\mathbf{x_i}^{out}) = \delta \tag{3.4}$$

Knowing that only surface points are available at first, it is necessary to find approximations of the normal vectors to generate the extra points. To that, two approaches are available: a)

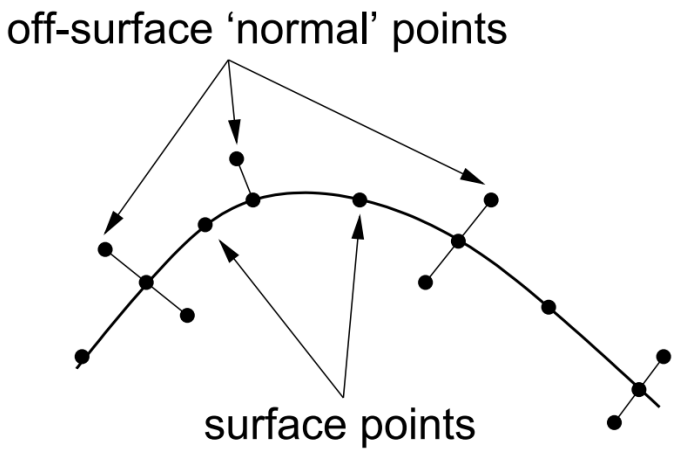


Figure 2 – Example of extra points added to the initial set of points. (CARR *et al.*, 2001)

get information from the scanner used to acquire the point cloud or b) make this approximation using a neighbourhood of the points as suggested by Hoppe *et al.* (1992). In case no normal vector can be generated for a specific point, no new points are added to it.

3.2.2 RBF Interpolation

Given all points $\{\mathbf{x_i}\}_{i=1}^{n} \subset \mathbb{R}^3$ and a set of values $\{f(\mathbf{x_i})\}_{i=1}^{n} \subset \mathbb{R}$, the goal is to find an interpolant $s(\mathbf{x_i}) = f_i$, $i = 1, ..., n$. This interpolant will be as follows:

$$s(\mathbf{x}) = p(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i \phi(|\mathbf{x} - \mathbf{x_i}|), \tag{3.5}$$

where $\mathbf{x_i}$ are called RBF **centers**, $p(\mathbf{x})$ is a low-degree polynomial, α_i are real coefficients, $||$ is the euclidean norm in $\mathbb{R}^3$ and $\phi(x)$ is a real function called Radial Basis Function (RBF).

It is also necessary to satisfy the following conditions of orthogonality: $\sum_{i=1}^{n} \alpha_i p_j(\mathbf{x_i}) = 0$.

The above equations can be written in matrix form as

$$\begin{pmatrix} A & P \\ P^T & O \end{pmatrix} \begin{pmatrix} \alpha \\ c \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}, \tag{3.6}$$

where

$$A = \begin{pmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{n1} & \phi_{n2} & \cdots & \phi_{nn} \end{pmatrix}, \tag{3.7}$$

GSRBF	$\phi(\mathbf{r})$	Continuity
Biharmonic (Linear)	$\phi(r) = r$	C^0
Triharmonic (Cubic)	$\phi(r) = r^3$	C^2
Gaussian	$\phi(r) = e^{-r^2/c^2}$	C^∞
Multiquadric (MQ)	$\phi(r) = (r^2 + c^2)^k, k > 0, k \notin \mathbb{N}$	C^∞
Inverse multiquadric (IMQ)	$\phi(r) = (r^2 + c^2)^{-k}, k > 0, k \notin \mathbb{N}$	C^∞
Thin plate spline (TPS)	$\phi(r) = \begin{cases} r^{2k-1}, & k \in \mathbb{N} \\ r^{2k} log(r), & k \in \mathbb{N} \end{cases}$	C^{2k}

Table 1 – Global supported radial basis functions. (MO; SHOU; CHEN, 2022)

with $\phi_{ij} = \phi(|\mathbf{x_i} - \mathbf{x_j}|)$, for $i, j = 1, 2, ..., n$, and

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1k} \\ p_{21} & p_{22} & \cdots & p_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nk} \end{pmatrix}, \tag{3.8}$$

with $p_{ij} = p_j(\mathbf{x_i})$, for $i = 1, 2, ..., n$, $j = 1, 2, ..., k$, and O is a zero matrix of dimensions $(k \times k)$.

There are several RBFs that can be used, which have been used in various works, some of which can be seen in Table 1.

By solving the linear system 3.6, the coefficients α and c can be defined, which leads to the computation of the interpolant $s(\mathbf{x})$. However, the direct methods to solve linear systems, like Cholesky Decompostion, are very time-consuming. Hence, using them is only feasible for an initial point set that contains at most a few thousands points. This problem will be solved by using methods presented in the next sections.

3.2.3 HRBF Interpolation

Constructing offset-points poses challenges as it demands parameter tuning, introducing a potential for errors in the process. In light of these difficulties, an alternative approach is presented here, the Hermite Radial Basis Functions (HRBF). The methodology detailed in this subsection is based on the work of Fasshauer (2007) and Macedo, Gois and Velho (2011).

Unlike the standard RBF formulation, for the Hermite Radial Basis Function Interpolation, the surface is reconstructed by using a set of points with their their normal vectors. So, a set of points $\{\mathbf{x_i}\}_{i=1}^{n} \subset \mathbb{R}^3$ and their normals $\{\mathbf{n_i}\}_{i=1}^{n} \subset \mathbb{R}^3$ are given and we're trying to find a function $s : \mathbb{R}^3 \to \mathbb{R}$ that satisfies both $s(x_i) = 0$ and $\nabla s(\mathbf{x_i}) = n_i$.

Theorem 3.2.1 (Riesz Representation Theorem)**.** For every linear functional $\lambda_i \in \mathbb{H}^*$, there is a unique vector $\mathbf{v_i} \in \mathbb{H}$, called Riesz representer, such that $\lambda_i(s) = \langle \mathbf{v_i}, s \rangle$, for all $s \in \mathbb{H}$.

What we aim to find is the norm-minimal generalized interpolant, denoted as s^*, which satisfies the following optimization problem:

$$||s^*|| = min\{||s|| : s \in \mathbb{H}, \lambda_i(s) = f_i, 1 \leq i \leq n\}, \tag{3.9}$$

where $\lambda_1, ..., \lambda_n \in \mathbb{H}$ are linearly independent functionals with Riesz representers $\mathbf{v_1}, ..., \mathbf{v_n}$ and $f_1, ..., f_n \in \mathbb{R}$ are given values. It is known that the unique solution for 3.9 is a linear combination of the Riesz representers:

$$s^* = \sum_{i=1}^{n} \alpha_i v_i, \tag{3.10}$$

where the coefficients $\{\alpha_i\}$ will be determined by the conditions $\lambda_i(s^*) = f_i, 1 \leq i \leq n$.

Denoting $\delta_{\mathbf{x}}(s) := s(\mathbf{x})$ as the evaluation functional at the point $\mathbf{x} \in \mathbb{R}^3$, we will have the following intepolating conditions:

$$\begin{aligned}
\lambda_i(s) &= \delta_{\mathbf{x_i}} = 0 \\
\lambda_i^x(s) &= \delta_{\mathbf{x_i}} \circ \frac{\partial}{\partial x}(s) = n_i^x \\
\lambda_i^y(s) &= \delta_{\mathbf{x_i}} \circ \frac{\partial}{\partial y}(s) = n_i^y \\
\lambda_i^z(s) &= \delta_{\mathbf{x_i}} \circ \frac{\partial}{\partial z}(s) = n_i^z.
\end{aligned} \tag{3.11}$$

Following the theory of Hermite-Birkhoff interpolation, what we need to construct is a Hilbert Space $\mathbb{H}$, such that these functionals are continuous and linearly independent and where both Riesz representers and inner products between them can be easily computed.

Actually, for the positive-definite radial basis functions $\phi : \mathbb{R}_+ \to \mathbb{R}$, such that $\Psi := \phi(||\cdot||) \in C^{2k}(\mathbb{R}^3) \cap L^1(\mathbb{R}^3)$, there is a native Hilbert Space $\mathbb{H}_\phi \subset C^k(\mathbb{R}^3)$. For this space, the

functionals from 3.12 are continuous and if they are pairwise distinct, they are also linearly independent. Their Riesz representers are given by:

$$\begin{aligned} v_i &= \Psi(\cdot - \mathbf{x}) \\ v_i^x &= -\frac{\partial}{\partial x}\Psi(\cdot - \mathbf{x}) \\ v_i^y &= -\frac{\partial}{\partial y}\Psi(\cdot - \mathbf{x}) \\ v_i^z &= -\frac{\partial}{\partial z}\Psi(\cdot - \mathbf{x}). \end{aligned} \tag{3.12}$$

Thus, the resulting minimum $\mathbb{H}$-norm generalized interpolant s^* has the form

$$s^*(\mathbf{x}) = \sum_{i=1}^{n} \{\alpha_i \Psi(\mathbf{x} - \mathbf{x_i}) - \langle \beta_\mathbf{i}, \nabla\Psi(\mathbf{x} - \mathbf{x_i})\rangle\}, \tag{3.13}$$

where $\alpha_i \in \mathbb{R}$ and $\beta_i \in \mathbb{R}^3$ are uniquely determined by the following conditions:

$$\begin{aligned} s^*(\mathbf{x_i}) &= \sum_{j=1}^{n} \{\alpha_j \Psi(\mathbf{x_i} - \mathbf{x_j}) - \langle \beta_\mathbf{j}, \nabla\Psi(\mathbf{x_i} - \mathbf{x_j})\rangle\} = 0 \\ \nabla s^*(\mathbf{x_i}) &= \sum_{j=1}^{n} \alpha_i \nabla\Psi(\mathbf{x_i} - \mathbf{x_j}) - H\Psi(\mathbf{x} - \mathbf{x_i})\beta_\mathbf{j}\} = \mathbf{n_i}, \end{aligned} \tag{3.14}$$

with H being the hessian operator of the second order partial derivatives.

3.3 2^n-tree

2^n-tree is a type of spatial partitioning data structure. A 2^n-tree represents the partitioning of a n-dimensional space into several regions in a hierarchical fashion. It is a tree in which each non-leaf node has exactly 2^n nodes. Each node is a region that comes from the sub-division of its parent node, except the root node, that represents the initial space. Two particular cases of this tree will be presented next: Quadtree(2^2-tree) and Octree(2^3-tree).

3.3.1 Quadtree

Quadtree is a particular case of the 2^n-tree, where $n = 2$. It was first presented by Finkel and Bentley (1974). The purpose of this data structure is to represent a 2D space by recursively

subdividing it at each step into four regions (quadrants). These regions can be squares, rectangles or another shape which are subdivided until a predetermined condition is reached. Each node of this tree has either four or zero nodes. In the latter, the node is a leaf, where the relevant data of the application are kept. The type of data kept in the data structure depends on the purpose of the application. They could be, for example, points, lines or curves. Figure 3 illustrates a Quadtree partitioning a set of points.

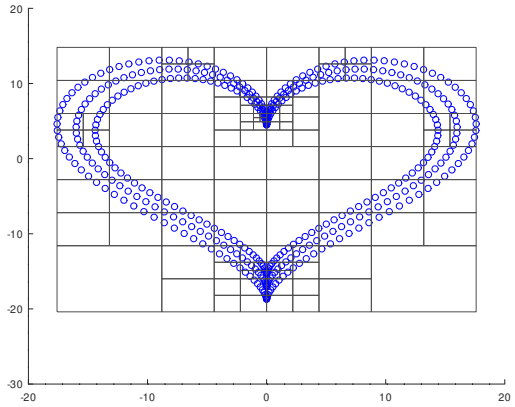


Figure 3 – Example of a point set partitioned with a Quadtree.

3.3.2 Octree

Octree is another particular case of the 2^n-tree, where $n = 3$. It is similar to Quadtree, but it suits the representation of 3D spaces. Here, at each step, the space is subdivided into 8 regions (octants). This data structure can be used to represent points, surfaces or volumes in 3D. An example of an Octree is shown in Figure 4. More information about this data structure can be found on Samet (1984), Samet (2006).

3.4 Multilevel Partition of Unity

One major problem with the RBF method described in section 3.2 is its global characteristic. That is, solving the linear systems for large point sets can be infeasible and, also, after interpolants are calculated, it is necessary to perform many operations to evaluate a certain point, given that there is a coefficient for each RBF center.

The Partition of Unity is a method that partitions the function domain into several subdomains and calculates an interpolant for each sub-domain created. This way, the linear systems needed to be solved are smaller and when it is necessary to evalute a point in the function, the result can be a mix of the interpolants of a close neighbourhood of that point.

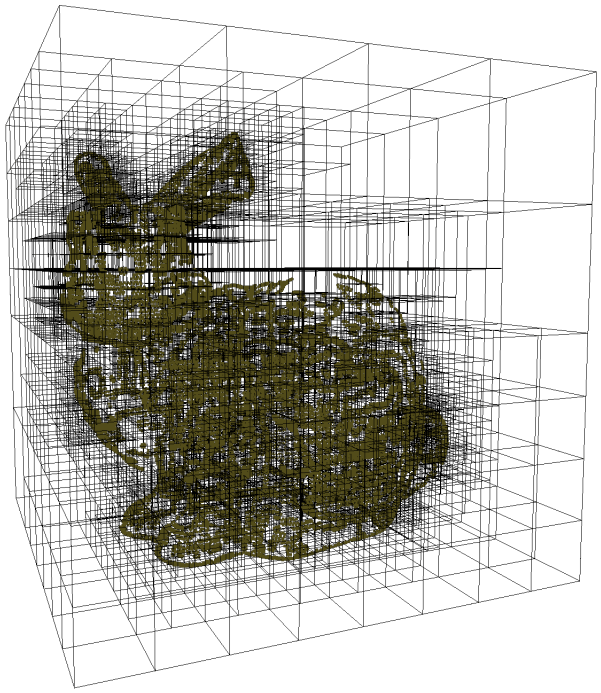


Figure 4 – Example of the Stanford Bunny point cloud partitioned with an Octree.

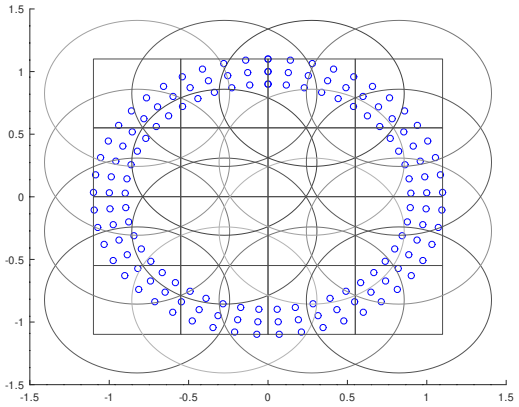


Figure 5 – Example of a partitioned domain and its supports.

3.4.1 Partition of Unity

Consider a bounded domain Ω in an Euclidian space and a set of functions Φ, where each of its elements Φ_i is relative to a sub-domain of Ω and $\sum_i \Phi_i = 1$.

Given the functions s_i that locally approximate the points belonging to each sub-domain, the global function that interpolates the points can be given by:

$$f(x) = \sum_i \Phi_i(x) s_i(x). \tag{3.15}$$

And given a set of non-negative functions that have compact support w_i such that

$\Omega \subset \bigcup_i supp(w_i)$, the partition of unity can be generated by

$$\Phi_i(x) = \frac{w_i(x)}{\sum_{j=1}^{n} w_j(x)}, \tag{3.16}$$

where the inverse-distance singular weights (FRANKE; NIELSON, 1980; RENKA, 1988) is going to be used as the weight function

$$w_i(x) = \left(\frac{(R_i - |x - c_i|)_+}{R_i |x - c_i|} \right)^2, \text{ where } (a)_+ = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

centered at c_i and having R_i as the support radius.

This way, when to evaluate a point in $f(x)$, the considered local interpolants s_i are those such that $|x - c_i| < R_i$. An illustration of the partition of unity in 2D can be seen in Figure 5.

For each sub-domain, the support radius R_i is initially given by $R_i = \alpha d$, where d is the cell diagonal. If the amount of points inside this radius is less than N_{min}, the radius is increased by setting $R'_i = R_i$ and iterating until it contains at least N_{min} points, as follows:

$$R'_i = R'_i + \lambda R_i. \tag{3.18}$$

This iterative process in illustrated in Figure 6.

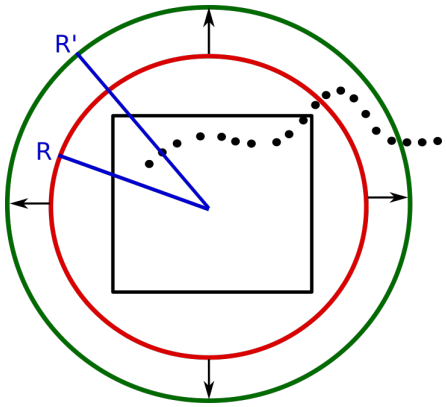


Figure 6 – Increasing the support radius of a sub-domain.

3.4.2 Adaptive Approximation based on Octree

The Octree data structure, presented in 3.3, is employed to make the domain partitioning. With this data structure, it is possible to make an adaptive partitioning of our domain, that is, it is

possible to have a higher refinement where there are more points or the surface complexity is higher.

The root node of the Octree represents a cube that bounds the initial set of points. Next, a stop condition for the partitioning is defined. That could be the amount of points inside the cell being small enough, a threshold of an error function that has been reached, etc. Then, the partitioning goes recursively as follows:

- If the defined stop condition hasn't been reached yet, then, for each of the eight octants, a new node is created as a child of the current node being processed.
- Else, this node becomes a leaf of the tree and a local interpolant is calculated for this sub-domain.

For a better understanding of the process, please refer to Figure 7.

After having a local interpolant calculated for each sub-domain, i.e. the Octree leafs, when it is necessary to evaluate a point in the global interpolant, we can only navigate through the tree going down to the nodes such that the space it represents has a support radius that contains the point. Then, we can mix the local interpolants results obtained in the leafs as described in the previous sub-section.

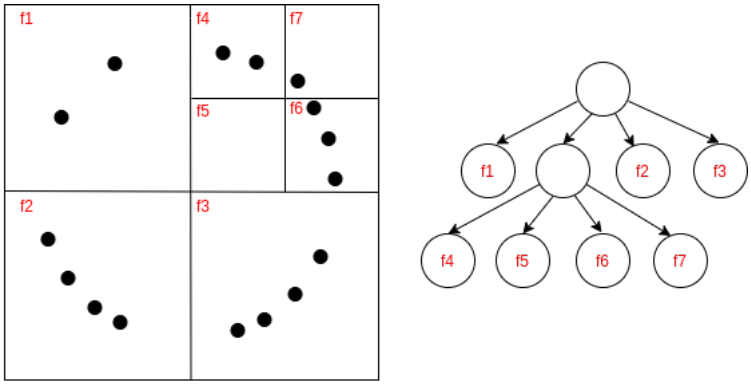


Figure 7 – Example of a 2D adaptive partitioning, where the stop condition is to have less than 5 points.

3.5 Marching Cubes

Marching Cubes is an algorithm, presented by Lorensen and Cline (1987), to extract triangular meshes from isosurfaces like those generated by the previous sections methods. The first step of this algorithm is to create a mesh grid of equally spaced points that bounds the input volume and then evaluating all points of this grid in the implicit function, detecting for each point if they are inside or outside the surface.

After knowing for each point if they are inside or outside the surface, the algorithm iterates over all cubes (sets of eight neighbour points) of the mesh grid and, for each one of

them, creates the triangles that will represent the part of the surface that passes through the cube. Finally, all generated triangles are mixed into the final triangular mesh.

To detect which triangles are going to be created for each cube, a mapping of possibilities is precalculated. This mapping tells which triangles should be created depending on the configuration of which points are inside and outside the surface. There are $2^8 = 256$ different possibilities of configurations for a given cube, because each point can be either inside or outside the surface. However, if the symmetries are not considered, there are only 15 different configurations, those are illustrated in Figure 8.

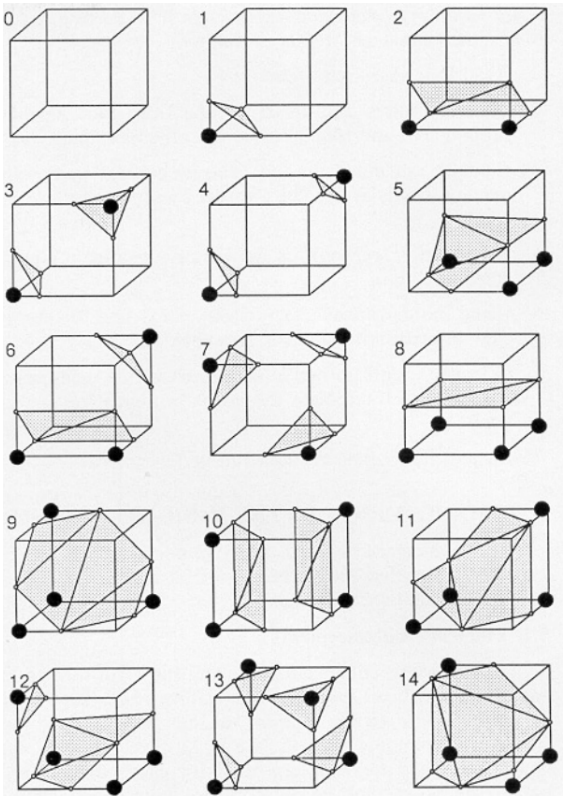


Figure 8 – Triangulated cubes. (LORENSEN; CLINE, 1987)

CHAPTER

4

MATERIALS AND METHODS

In this chapter we will introduce our strategy to solve the surface reconstruction problem based on the methods presented in Chapter 3. First, we present the dataset used in the experiments. Next, we show how we obtain the interpolant using Multilevel Partition of Unity with HRBF. Finally, we demonstrate how the Marching Cubes algorithm was applied to extract the triangular mesh from the implicit function obtained in the previous methods. Figure 9 depicts the pipeline of our proposed method.

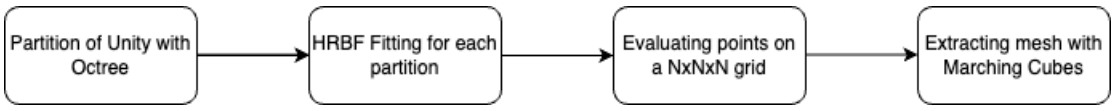


Figure 9 – Workflow diagram

The surface reconstruction methods and mesh extraction were all implemented using the Python programming language. HRBF interpolation and Multilevel Partition of Unity implementation included the numerical and scientific libraries NumPy and SciPy, the Marching Cubes implementation we used came from the Scikit-image library and the tool to visualize the generated surfaces was MeshLab, an open source system for processing and editing triangular meshes.

4.1 Dataset

The dataset employed in this work is the 2D/3D Face Dataset from The Media Integration and Communication Center (MICC) and University of Florence (BAGDANOV; BIMBO; MASI, 2011). This dataset was built specifically for research in various face analysis and recognition works. It consists of high-resolution 3D scans of human faces from several subjects and some videos on different conditions, zoom level and resolution, as illustrated in Figure 10.

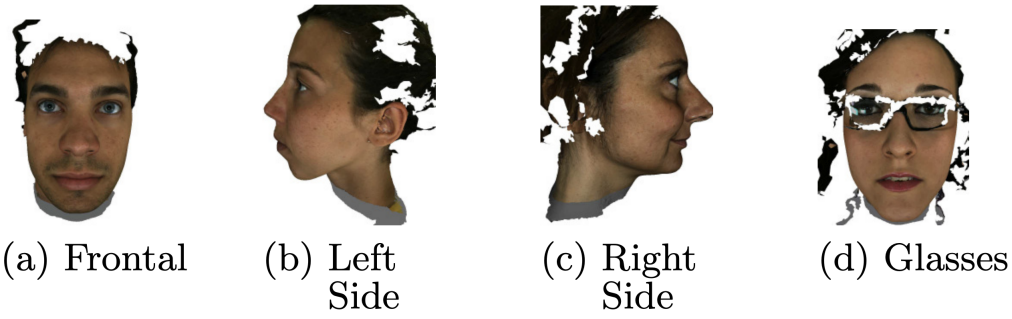


Figure 10 – Examples of 3D models of Florence dataset. (BAGDANOV; BIMBO; MASI, 2011)

The 2D/3D Florence Face Dataset consisted of 53 subjects at the time of our access. However, the authors have indicated that it is a work in progress, suggesting the possibility of additional subjects in the future. Each subject has four distinct scans, consisting of two frontal and two side views. In our project, we specifically used 27 subjects, selecting only one frontal scan per subject. The models have around 40,000 vertices and 80,000 facets in average and are accurate to 0.2mm. The details regarding the number of vertices and faces for the utilized models can be referenced in Table 2.

4.2 Multilevel Partition of Unity

We utilize the Multilevel Partition of Unity based on Octree with HRBF to derive the implicit function defining the surface approximation, as detailed in Section 3.4.

The initial step involves identifying a cube that encapsulates all points within the initial set, serving as the root node of the Octree. Recursively, we partition this space into 8 octants until the defined stopping conditions are met. To illustrate, the figure 11 shows the initial four levels of the Octree partition for a point cloud.

The stopping conditions for partitioning are twofold: a) having 100 points, or fewer inside the cell or b) the tree depth reaching a maximum of 12. Upon meeting these criteria, the node becomes a leaf in the tree, requiring the calculation of the local interpolant.

We initialize the support radius of a sub-domain as $R_i = d$, where d is the cell diagonal. If the number of points inside the cell is less than N_{min}, we increment its support radius by setting $R'_i = R_i$ and iterate, as mentioned in Section 3.4, until there are at least N_{min} points inside it:

$$R'_i = R'_i + \lambda R_i. \tag{4.1}$$

Here, we set $N_{min} = 15$ and $\lambda = 0.1$, as in Ohtake *et al.* (2003).

We employed a K-D Tree to determine the number of points enclosed by a given sphere.

Dataset	# of points in the original mesh	# of faces in the original mesh
Face 1	39694	77868
Face 2	58390	114865
Face 3	48813	96635
Face 4	39115	77800
Face 5	47237	93791
Face 6	50401	99581
Face 7	45808	90224
Face 8	48445	96310
Face 9	47947	94196
Face 10	26629	52758
Face 11	45260	89869
Face 12	43950	86711
Face 13	46810	92176
Face 14	45730	88155
Face 15	35234	69940
Face 16	50730	100192
Face 17	44487	87495
Face 18	44093	87575
Face 19	46327	90693
Face 20	45798	90784
Face 21	45796	90765
Face 22	47318	93033
Face 23	44178	87504
Face 24	37349	73553
Face 25	43669	85986
Face 26	27567	52604
Face 27	36836	72912

Table 2 – Number of points and faces in the original meshes.

4.3 HRBF Interpolation

Once the cell and the size of the support radius are established, we compute an HRBF Interpolation for this set of points. Our choice for the Radial Basis Function (RBF) is $\Psi(\mathbf{x}) := \phi(||\mathbf{x}||) = ||\mathbf{x}||^3$. By solving the linear system derived from these points, we obtain the local interpolant for the leaf node. The Python NumPy library was used to solve linear systems. It is based on LAPACK, more specifically the Cholesky decomposition.

It is worth noting that the fitting time can be enhanced by means of parallelism, as the linear systems of each Octree node can be solved concurrently.

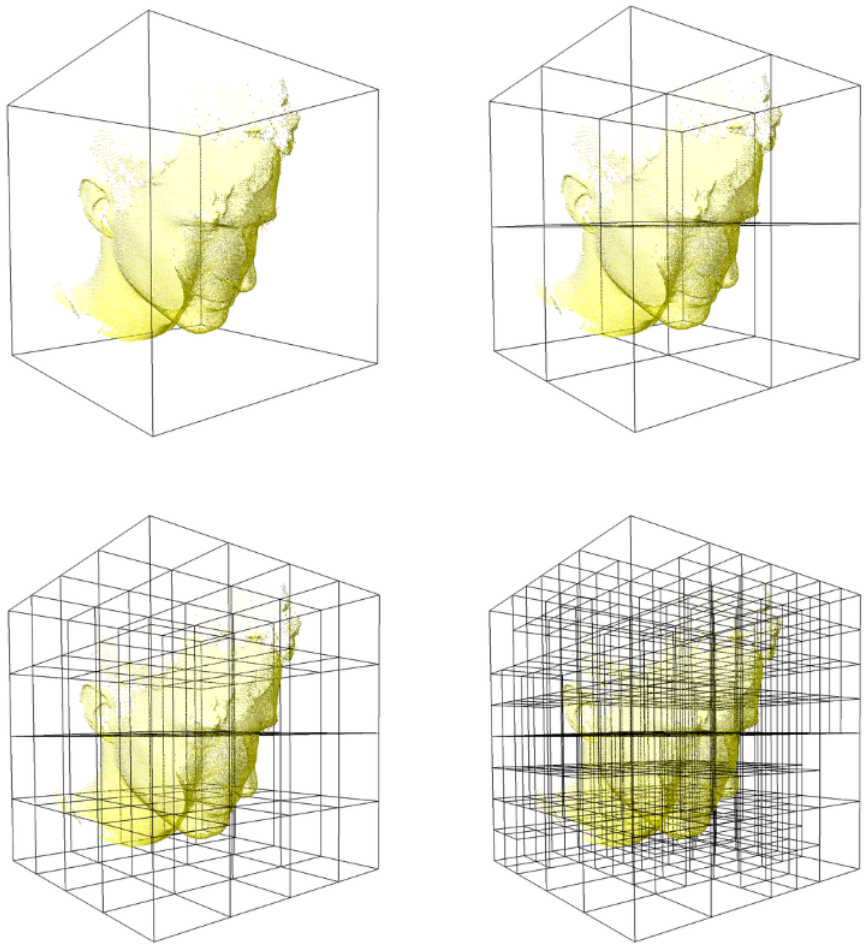


Figure 11 – Octree partitioning in 4 levels

4.4 Evaluating points on a grid

After obtaining the local interpolants calculated for each node of the Octree, the next step involves evaluating them on an *NxNxN* grid for subsequent triangular mesh extraction. In our work, we generated a $150x150x150$ grid for this purpose.

However, once the grid is established, we selectively compute the values for points that are near the surface. During the evaluation of a point, we assess whether its distance to the point cloud is within 6 times the size of the cube cell edge of the mesh grid. This criterion allows us to focus computation efforts on points that are close enough to the surface, optimizing the efficiency of our process. In order to efficiently determine the distance to the closest point in the point cloud, we make use of a K-D Tree.

To evaluate the function at a point x, we gather all pertinent local interpolants by traversing the Octree from the root node downwards, selecting nodes where the support radius bounds the point, i.e., $|x - c_i| < R_i$.

4.5 Extracting mesh with Marching Cubes

Now that we have the implicit function that interpolates the surface, the next step involves extracting the triangular mesh from this function for visualization.

For this, we used the Marching Cubes algorithm, as discussed in 3.5. The process begins with the mesh grid that was generated in the previous step. Hence, we can determine which points are either inside or outside the surface. This information serves as input to the Marching Cubes algorithm, that will generate the triangular mesh.

We used the the Marching Cubes implementation from Scikit-image. The resulting meshes are saved in the OBJ format and , hence, easily visualized with MeshLab.

CHAPTER

5

RESULTS

In this chapter we present the results obtained by our method from the Florence 2D/3D Face Dataset described in 4.1. The tests were conducted on an Apple Macbook Air 2020 computer, with M1 processor and 16GB RAM, with the macOS 13.5.2 operating system.

We have selected 13 models and presented the reconstructions to show the strengths and the pitfalls of our strategy. For each model we show (a) the original mesh with texture, (b) the original mesh without texture, (c) the point cloud extracted from the original mesh and, finally, (d) the mesh obtained from our reconstruction.

For certain faces, the method demonstrated the ability to fill in some gaps in the surfaces. In the model shown in 12, 13, 16, 21, 22, 23 and 24 the method was capable of reconstruct missing parts of the hair. The model in 19 was reconstructed with the missing parts of both the hair and beard.

In the depicted surface of Figure 21, a noticeable area with sparse data appears in the head. But although the method filled the gap, it is evident that the reconstruction in this specific region lacks precision.

However, in some cases, we observed that the method generated surfaces external to the main surface. This is primarily attributed to point clouds in complex or data-sparse areas. Figures 18, 23, and 24 highlight this problem in the hair region. While in 20, the issue becomes evident in the beard region.

The analysis of the resulting data revealed that the implementation of an Octree structure led to a significant improvement in both memory efficiency and computational time. This was primarily due to the ability of the Octree to effectively partition the point cloud data into smaller, more manageable subsets, thereby reducing the complexity of the surface reconstruction process. Furthermore, the choice of mesh grid size for the surface reconstruction process resulted in a high-resolution mesh, which in turn captured a greater level of detail on the surface.

Dataset	**# of points on resulting mesh**	**# of facets on resulting mesh**	**Fitting time (sec)**	**Evaluation time (sec)**	**Memory peak (MB)**	**# of nodes in octree**	**Avg # of points in octree node**
Face 1	104694	209208	80.08	278.63	140.41	2918	42.97
Face 2	97690	195042	101.34	264.47	160.85	5477	34.84
Face 3	107646	215002	102.60	315.26	150.64	4297	37.01
Face 4	107127	214118	86.39	296.99	138.92	2559	46.95
Face 5	113017	225848	107.96	308.27	149.12	4163	36.92
Face 6	94223	188340	89.94	275.15	153.27	4819	34.48
Face 7	106321	212570	83.89	280.11	148.64	4344	34.84
Face 8	106521	212908	98.60	118.60	149.17	3837	40.07
Face 9	99485	198784	77.10	268.55	151.19	4759	33.57
Face 10	111543	222872	44.02	286.85	128.33	2099	39.86
Face 11	107000	213780	81.56	295.75	147.44	4083	36.13
Face 12	119846	239510	90.71	314.63	144.62	3424	40.64
Face 13	111161	222134	93.36	296.73	149.03	4229	36.30
Face 14	118848	237526	81.81	83.27	149.21	4524	34.19
Face 15	112682	225226	93.15	303.46	135.39	2302	46.75
Face 16	96939	193644	111.27	295.92	153.58	4861	34.60
Face 17	101108	202056	98.58	278.62	147.23	4250	34.59
Face 18	111819	223430	105.10	315.10	145.34	3625	38.96
Face 19	99190	198102	93.31	278.14	149.16	4450	34.34
Face 20	107843	215526	111.22	313.80	146.92	3762	38.70
Face 21	105453	210750	98.66	295.66	148.35	4271	35.22
Face 22	117196	234232	100.36	308.83	148.54	3939	38.57
Face 23	113856	227420	94.59	331.34	144.12	3103	44.52
Face 24	113992	227700	81.84	304.49	137.14	2340	48.92
Face 25	133957	267628	91.30	326.81	144.69	3487	40.04
Face 26	104707	209302	64.11	267.63	128.46	1828	46.85
Face 27	119009	237826	76.21	303.85	136.62	2361	47.28

Table 3 – Perfomance metrics of our method.

Details, including the number of points and facets on the resulting mesh, fitting time, evaluation time, peak memory usage, number of nodes in the octree, and average number of points inside an octree node, are provided in Table 3.

(a) Original mesh with texture.

(b) Original mesh without texture.

(c) Point cloud extracted from original mesh.

(d) Mesh obtained from our reconstruction.

Figure 12 – Face 1 result

(a) Original mesh with texture.

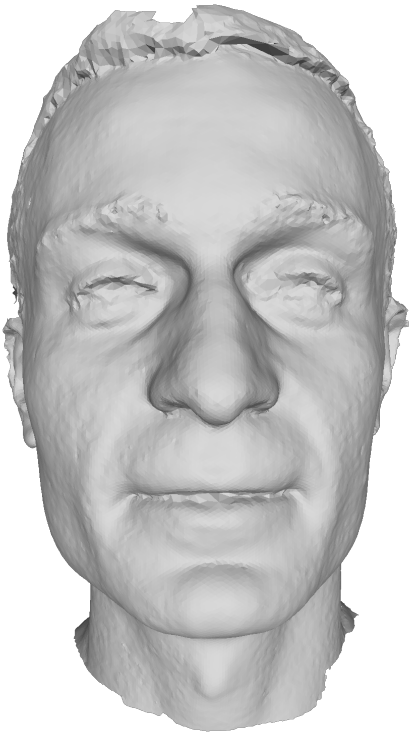
(b) Original mesh without texture.

(c) Point cloud extracted from original mesh.

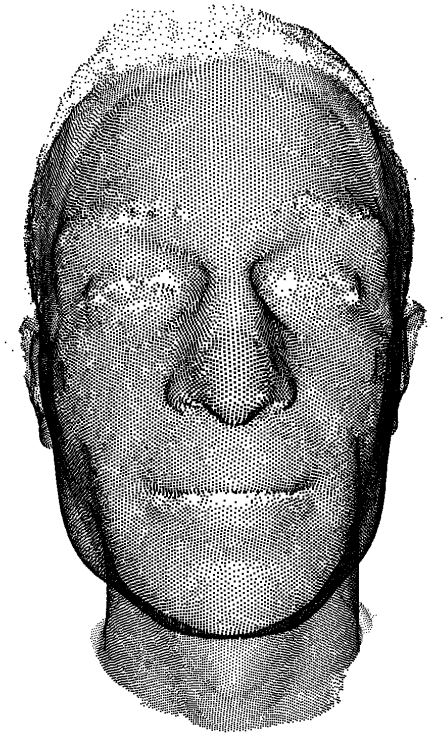
(d) Mesh obtained from our reconstruction.

Figure 13 – Face 2 result

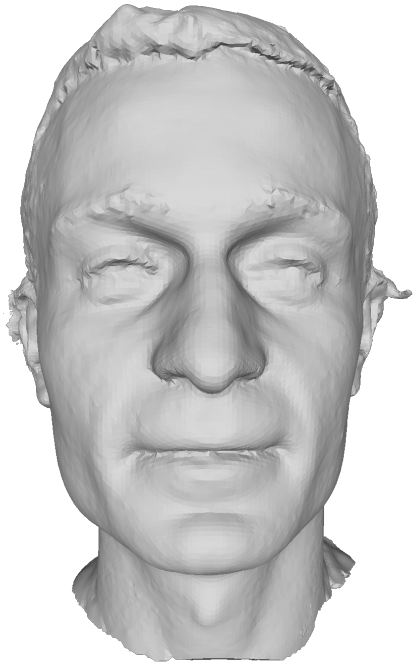
(a) Original mesh with texture.



(b) Original mesh without texture.



(c) Point cloud extracted from original mesh.

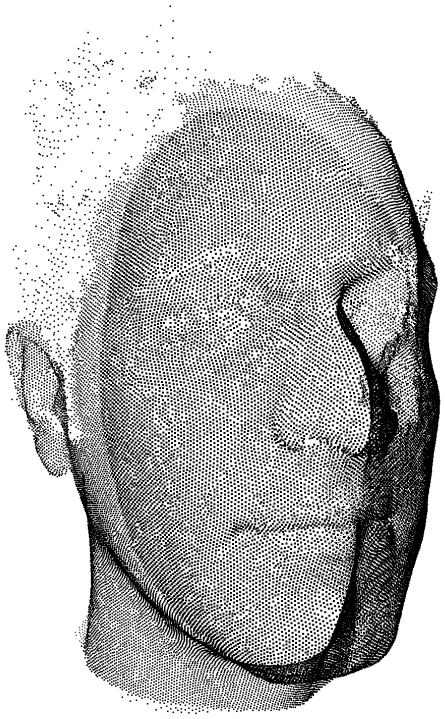


(d) Mesh obtained from our reconstruction.

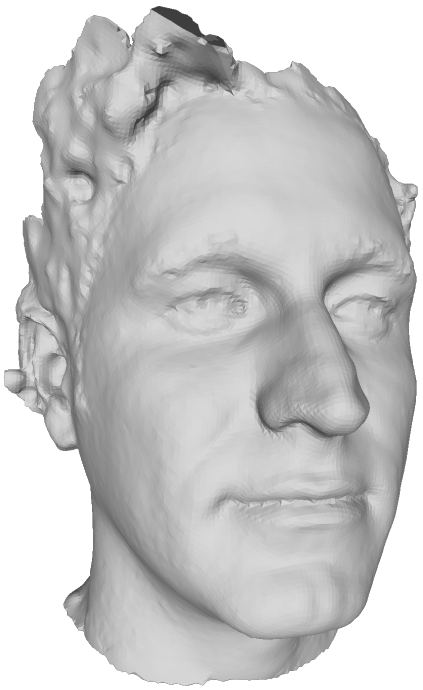
Figure 14 – Face 3 result

(a) Original mesh with texture.

(b) Original mesh without texture.



(c) Point cloud extracted from original mesh.



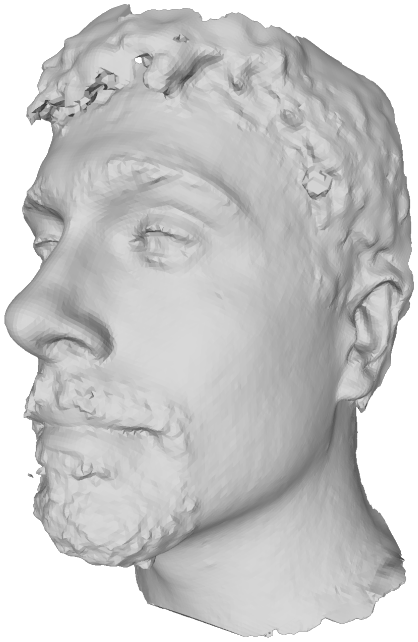
(d) Mesh obtained from our reconstruction.

Figure 15 – Face 4 result

(a) Original mesh with texture.

(b) Original mesh without texture.

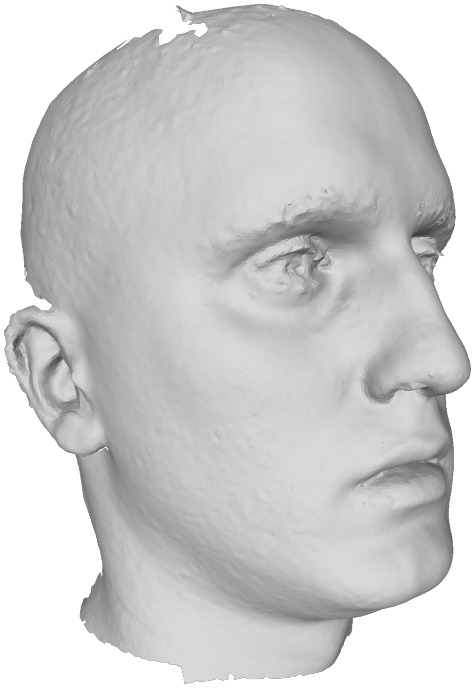
(c) Point cloud extracted from original mesh.



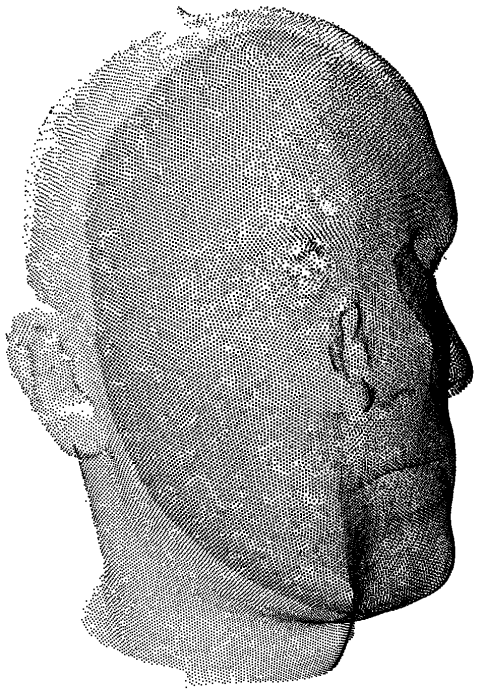
(d) Mesh obtained from our reconstruction.

Figure 16 – Face 5 result

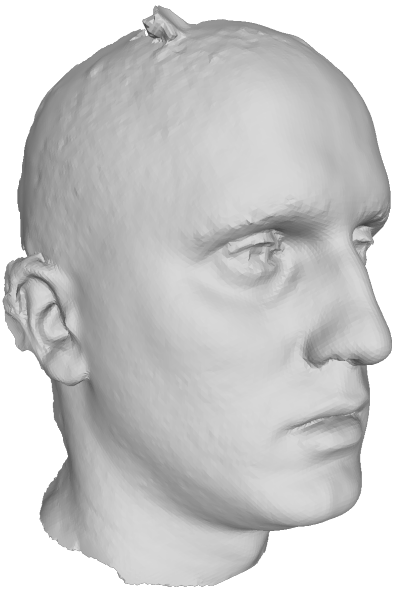
(a) Original mesh with texture.



(b) Original mesh without texture.



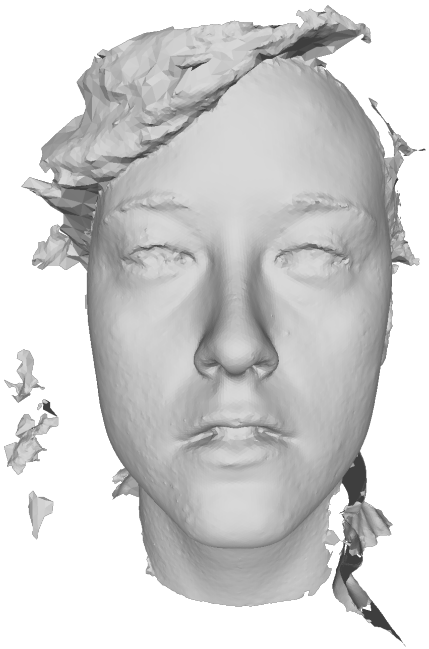
(c) Point cloud extracted from original mesh.



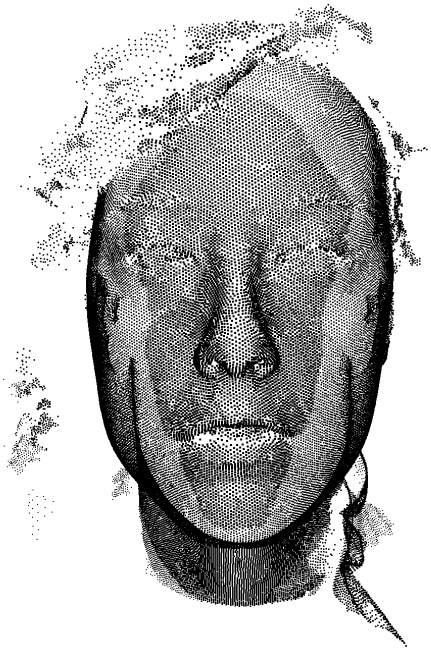
(d) Mesh obtained from our reconstruction.

Figure 17 – Face 8 result

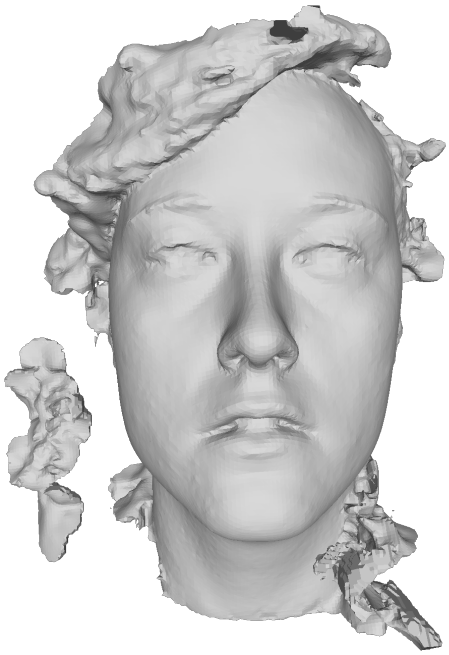
(a) Original mesh with texture.



(b) Original mesh without texture.



(c) Point cloud extracted from original mesh.



(d) Mesh obtained from our reconstruction.

Figure 18 – Face 12 result

(a) Original mesh with texture.

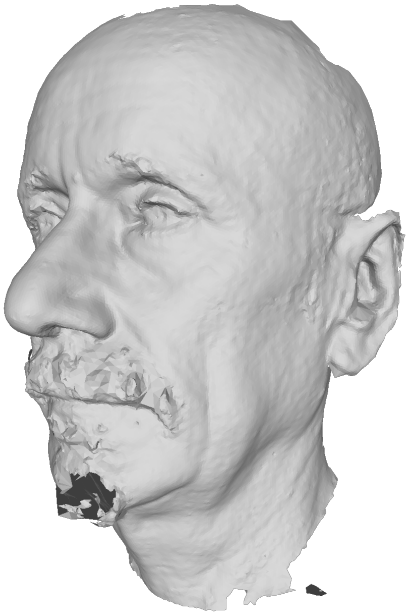
(b) Original mesh without texture.

(c) Point cloud extracted from original mesh.

(d) Mesh obtained from our reconstruction.

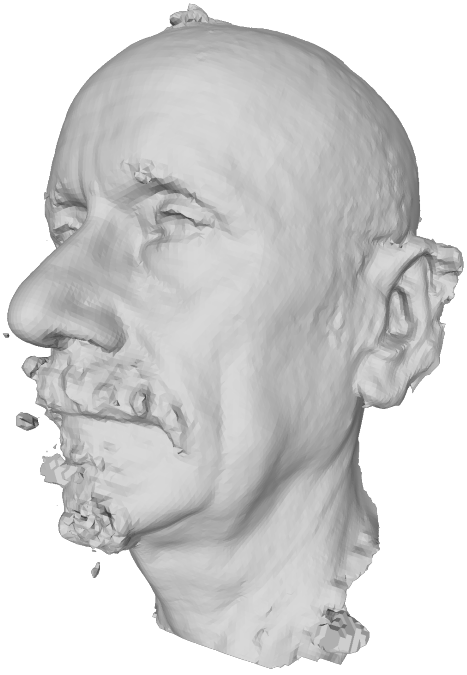
Figure 19 – Face 14 result

(a) Original mesh with texture.



(b) Original mesh without texture.

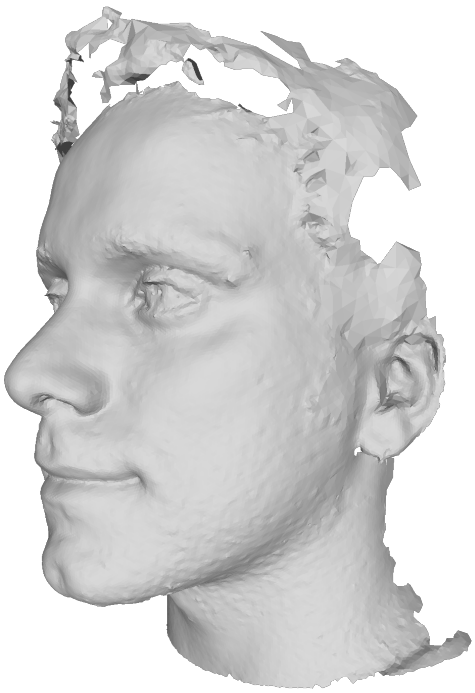
(c) Point cloud extracted from original mesh.



(d) Mesh obtained from our reconstruction.

Figure 20 – Face 16 result

(a) Original mesh with texture.



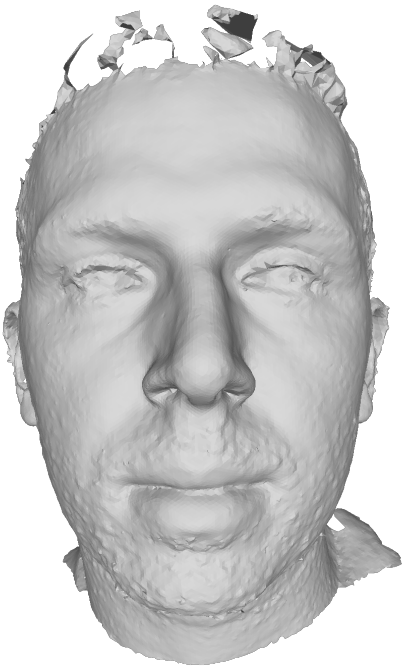
(b) Original mesh without texture.

(c) Point cloud extracted from original mesh.

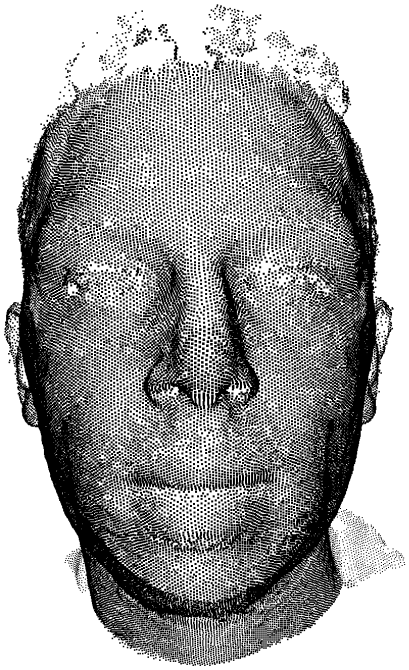
(d) Mesh obtained from our reconstruction.

Figure 21 – Face 17 result

(a) Original mesh with texture.



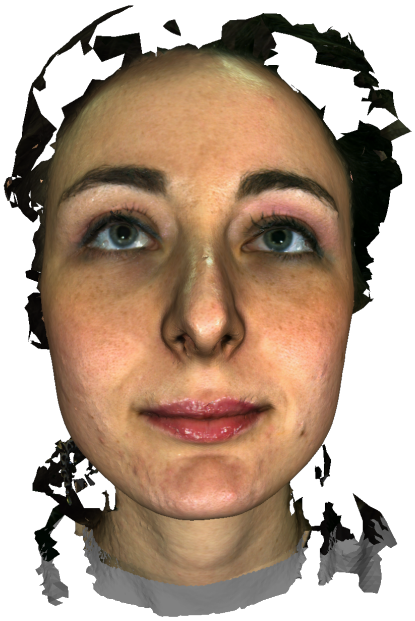
(b) Original mesh without texture.



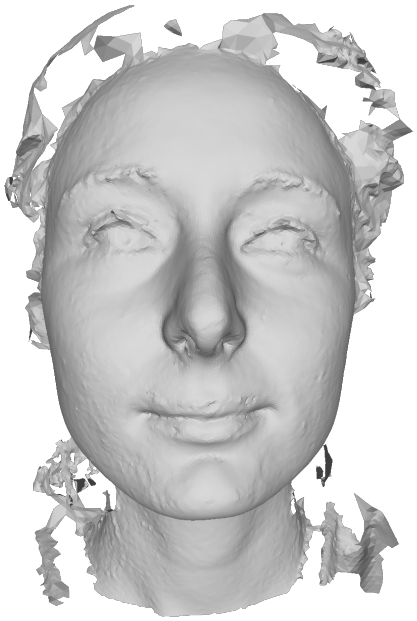
(c) Point cloud extracted from original mesh.

(d) Mesh obtained from our reconstruction.

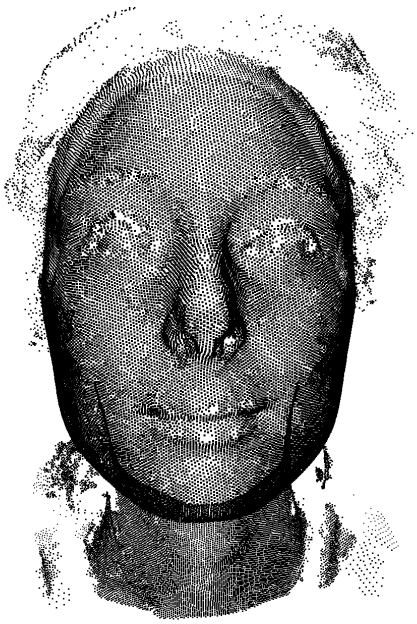
Figure 22 – Face 20 result



(a) Original mesh with texture.



(b) Original mesh without texture.



(c) Point cloud extracted from original mesh.

(d) Mesh obtained from our reconstruction.

Figure 23 – Face 25 result

(a) Original mesh with texture.

(b) Original mesh without texture.

(c) Point cloud extracted from original mesh.

(d) Mesh obtained from our reconstruction.

Figure 24 – Face 26 result

CHAPTER

6

CONCLUSION

6.1 Conclusion

In this work, we presented an approach for surface reconstruction from point clouds that combines the advantages of Hermite Radial Basis Function Implicits (HRBF) and the Multilevel Partition of Unity (MPU) method, such as proposed by Macedo, Gois and Velho (2011). Throughout the work, we covered all the necessary background theoretical topics for the development of this project, including Radial Basis Function (RBF) and Hermite Radial Basis Function (HRBF) interpolation, 2^n-tree (and more specifically, Quadtree and Octree), Multilevel Partition of Unity, and Marching Cubes.

Since our work is part of the FAPESP project "Mapeamento de características robustas entre diferentes domínios e espaços R^2 e R^3", which includes other works related to faces, for testing, we used the 2D/3D Florence Face Dataset. Our approach is able to accurately reconstruct facial geometry with details within a reasonable time and memory consumption, making it suitable for large point clouds.

As future work, there are several paths for exploration. The method can be subjected to experimentation with different stop conditions for Octree partitioning, other weight functions, and alternative Radial Basis Functions (RBF). Testing the method with different point clouds, including variations in size and shape, will help us understand its versatility and robustness. The method can also be implemented in a distributed manner to improve efficiency, as we can solve the linear systems of each partition of the Octree in parallel.

BIBLIOGRAPHY

BACHMAN, G.; NARICI, L. **Functional analysis**. [S.l.]: Courier Corporation, 2000. Citation on page 25.

BAGDANOV, A. D.; BIMBO, A. D.; MASI, I. The florence 2d/3d hybrid face dataset. In: **Proceedings of the 2011 Joint ACM Workshop on Human Gesture and Behavior Understanding**. New York, NY, USA: ACM, 2011. (J-HGBU '11), p. 79–80. ISBN 978-1-4503-0998-1. Available: <http://doi.acm.org/10.1145/2072572.2072597>. Citations on pages 11, 18, 37, and 38.

BATAGELO, H. C.; GOIS, J. P. Least-squares hermite radial basis functions implicits with adaptive sampling. In: **Proceedings of Graphics Interface 2013**. [S.l.: s.n.], 2013. p. 109–116. Citation on page 23.

BERGER, M.; TAGLIASACCHI, A.; SEVERSKY, L. M.; ALLIEZ, P.; GUENNEBAUD, G.; LEVINE, J. A.; SHARF, A.; SILVA, C. T. A survey of surface reconstruction from point clouds. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [S.l.], 2017. v. 36, n. 1, p. 301–329. Citation on page 17.

BRAZIL, E. V.; MACEDO, I.; SOUSA, M. C.; VELHO, L.; FIGUEIREDO, L. H. D. Shape and tone depiction for implicit surfaces. **Computers & Graphics**, Elsevier, v. 35, n. 1, p. 43–53, 2011. Citation on page 22.

BUHMANN, M. D. Radial functions on compact support. **Proceedings of the Edinburgh Mathematical Society**, Cambridge University Press, v. 41, n. 1, p. 33–46, 1998. Citation on page 22.

CARR, J. C.; BEATSON, R. K.; CHERRIE, J. B.; MITCHELL, T. J.; FRIGHT, W. R.; MCCALLUM, B. C.; EVANS, T. R. Reconstruction and representation of 3d objects with radial basis functions. In: **Proceedings of the 28th annual conference on Computer graphics and interactive techniques**. [S.l.: s.n.], 2001. p. 67–76. Citations on pages 11, 17, 22, and 28.

CARR, J. C.; FRIGHT, W. R.; BEATSON, R. K. Surface interpolation with radial basis functions for medical imaging. **IEEE transactions on medical imaging**, IEEE, v. 16, n. 1, p. 96–107, 1997. Citations on pages 17 and 21.

CASTELO, A.; NONATO, L. G.; SIQUEIRA, M. F.; MINGHIM, R.; TAVARES, G. The j1a triangulation: An adaptive triangulation in any dimension. **Computers & Graphics**, Elsevier, v. 30, n. 5, p. 737–753, 2006. Citation on page 23.

DRAKE, K. P.; FUSELIER, E. J.; WRIGHT, G. B. Implicit surface reconstruction with a curl-free radial basis function partition of unity method. **SIAM Journal on Scientific Computing**, SIAM, v. 44, n. 5, p. A3018–A3040, 2022. Citation on page 23.

FASSHAUER, G. E. **Meshfree approximation methods with MATLAB**. [S.l.]: World Scientific, 2007. Citations on pages 17 and 29.

FINKEL, R. A.; BENTLEY, J. L. Quad trees a data structure for retrieval on composite keys. **Acta informatica**, Springer, v. 4, n. 1, p. 1–9, 1974. Citation on page 31.

FRANKE, R.; NIELSON, G. Smooth interpolation of large sets of scattered data. **International journal for numerical methods in engineering**, Wiley Online Library, v. 15, n. 11, p. 1691–1704, 1980. Citation on page 34.

GOIS, J. P.; POLIZELLI-JUNIOR, V.; ETIENE, T.; TEJADA, E.; CASTELO, A.; NONATO, L. G.; ERTL, T. Twofold adaptive partition of unity implicits. **The Visual Computer**, Springer, v. 24, n. 12, p. 1013–1023, 2008. Citation on page 23.

GOIS, J. P.; TREVISAN, D. F.; BATAGELO, H. C.; MACÊDO, I. Generalized hermitian radial basis functions implicits from polygonal mesh constraints. **The Visual Computer**, Springer, v. 29, p. 651–661, 2013. Citation on page 23.

GREENGARD, L.; ROKHLIN, V. A fast algorithm for particle simulations. **Journal of computational physics**, Elsevier, v. 73, n. 2, p. 325–348, 1987. Citation on page 22.

HOPPE, H.; DEROSE, T.; DUCHAMP, T.; MCDONALD, J.; STUETZLE, W. Surface reconstruction from unorganized points. In: **Proceedings of the 19th annual conference on computer graphics and interactive techniques**. [S.l.: s.n.], 1992. p. 71–78. Citations on pages 21 and 28.

LIMA, E. L. Variedades diferenciáveis. **Brasil: Instituto de Matemática**, 2011. Citation on page 25.

LIU, S.; XIAO, J.; HU, L.; LIU, X. Implicit surfaces from polygon soup with compactly supported radial basis functions. **The Visual Computer**, Springer, v. 34, p. 779–791, 2018. Citation on page 22.

LORENSEN, W. E.; CLINE, H. E. Marching cubes: A high resolution 3d surface construction algorithm. **ACM siggraph computer graphics**, ACM New York, NY, USA, v. 21, n. 4, p. 163–169, 1987. Citations on pages 11, 17, 35, and 36.

LU, D.; ZHAO, H.; JIANG, M.; ZHOU, S.; ZHOU, T. A surface reconstruction method for highly noisy point clouds. In: SPRINGER. **International Workshop on Variational, Geometric, and Level Set Methods in Computer Vision**. [S.l.], 2005. p. 283–294. Citation on page 17.

MACEDO, I.; GOIS, J. P.; VELHO, L. Hermite radial basis functions implicits. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [S.l.], 2011. v. 30, n. 1, p. 27–42. Citations on pages 18, 22, 29, and 59.

MO, J.; SHOU, H.; CHEN, W. Implicit surface reconstruction via rbf interpolation: A review. **Recent Patents on Engineering**, Bentham Science Publishers, v. 16, n. 5, p. 49–66, 2022. Citations on pages 13, 21, and 29.

MORSE, B. S.; YOO, T. S.; CHEN, D. T.; RHEINGANS, P.; SUBRAMANIAN, K. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In: IEEE COMPUTER SOCIETY. **Shape Modeling and Applications, International Conference on**. [S.l.], 2001. p. 0089–0089. Citation on page 22.

MORSE, B. S.; YOO, T. S.; RHEINGANS, P.; CHEN, D. T.; SUBRAMANIAN, K. R. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In: **ACM SIGGRAPH 2005 Courses**. [S.l.: s.n.], 2005. p. 78–es. Citation on page 17.

OHTAKE, Y.; BELYAEV, A.; ALEXA, M.; TURK, G.; SEIDEL, H. Multi-level partition of unity implicits. **ACM SIGGRAPH 2003 Papers**, 2003. Citations on pages 17, 23, and 38.

OHTAKE, Y.; BELYAEV, A.; SEIDEL, H.-P. A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions. In: IEEE. **2003 Shape Modeling International.** [S.l.], 2003. p. 153–161. Citation on page 22.

____. Sparse surface reconstruction with adaptive partition of unity and radial basis functions. **Graphical Models**, Elsevier, v. 68, n. 1, p. 15–24, 2006. Citation on page 23.

RENKA, R. J. Multivariate interpolation of large sets of scattered data. **ACM Transactions on Mathematical Software (TOMS)**, ACM New York, NY, USA, v. 14, n. 2, p. 139–148, 1988. Citation on page 34.

ROMEIRO, R.; MARROQUIM, R.; ESPERANÇA, C.; BREDA, A.; FIGUEREDO, C. M. Forensic facial reconstruction using mesh template deformation with detail transfer over hrbf. In: IEEE. **2014 27th SIBGRAPI Conference on Graphics, Patterns and Images**. [S.l.], 2014. p. 266–273. Citations on pages 11 and 23.

SAMET, H. The quadtree and related hierarchical data structures. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 16, n. 2, p. 187–260, 1984. Citation on page 32.

____. **Foundations of multidimensional and metric data structures**. [S.l.]: Morgan Kaufmann, 2006. Citation on page 32.

SAVCHENKO, V. V.; PASKO, A. A.; OKUNEV, O. G.; KUNII, T. L. Function representation of solids reconstructed from scattered surface points and contours. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [S.l.], 1995. v. 14, n. 4, p. 181–188. Citation on page 21.

TOBOR, I.; REUTER, P.; SCHLICK, C. Multi-scale reconstruction of implicit surfaces with attributes from large unorganized point sets. In: IEEE. **Proceedings Shape Modeling Applications, 2004.** [S.l.], 2004. p. 19–30. Citation on page 22.

TREVISAN, D. F.; GOIS, J. P.; BATAGELO, H. C. A low-cost-memory cuda implementation of the conjugate gradient method applied to globally supported radial basis functions implicits. **Journal of Computational Science**, Elsevier, v. 5, n. 5, p. 701–708, 2014. Citation on page 23.

TURK, G.; O'BRIEN, J. F. Modelling with implicit surfaces that interpolate. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 21, n. 4, p. 855–873, 2002. Citation on page 17.

TURK, G.; O'BRIEN, J. F. Variational implicit surfaces. **Georgia Institute of Technology**, 1999. Citation on page 21.

WENDLAND, H. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. **Advances in computational Mathematics**, Springer, v. 4, p. 389–396, 1995. Citations on pages 17 and 22.

____. **Scattered data approximation**. [S.l.]: Cambridge university press, 2004. Citation on page 22.

WU, Z. Compactly supported positive definite radial functions. **Advances in computational mathematics**, Springer, v. 4, p. 283–292, 1995. Citation on page 17.


ICMC
USP
SÃO CARLOS