UNIVERSIDADE FEDERAL FLUMINENSE

RAFAEL MARICATO MUSMANNO

# ESTIMAÇÃO DE NORMAIS EM NUVENS DE PONTOS PARA RECONSTRUÇÃO DE SUPERFÍCIES 3D COM FUNÇÕES DE BASE RADIAL

NITERÓI 2016

### UNIVERSIDADE FEDERAL FLUMINENSE

### RAFAEL MARICATO MUSMANNO

## ESTIMAÇÃO DE NORMAIS EM NUVENS DE PONTOS PARA RECONSTRUÇÃO DE SUPERFÍCIES 3D COM FUNÇÕES DE BASE RADIAL

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Gráfica.

Orientador: ANSELMO ANTUNES MONTENEGRO

> NITERÓI 2016

### RAFAEL MARICATO MUSMANNO

### ESTIMAÇÃO DE NORMAIS EM NUVENS DE PONTOS PARA RECONSTRUÇÃO DE SUPERFÍCIES 3D COM FUNÇÕES DE BASE RADIAL

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Gráfica.

Aprovada em Julho de 2016.

#### BANCA EXAMINADORA

Prof. Dr. Anselmo Antunes Montenegro - Orientador, UFF

Prof. Dr. Marcos Lage, UFF

Prof. Dr. Waldemar Celes, PUC-RJ

Niterói 2016

## Resumo

A geração de malhas a partir de nuvens de pontos é um problema bastante investigado em Computação Gráfica e Modelagem Geométrica. Um dos métodos mais utilizados é o que toma como entrada pontos munidos de vetores normais. Uma das maiores dificuldades na reconstrução de nuvens de pontos está exatamente em como estimar normais para configurações de nuvens arbitrárias, incluindo nuvens com muitos pontos, com amostragem irregular, com ruído e com topologia complexa, por exemplo, com vizinhanças tubulares em diferentes escalas.

Este trabalho de pesquisa investiga uma nova forma de reconstruir malhas via nuvens de pontos. Neste novo tipo de reconstrução de malhas, normais são estimadas usando uma estrutura hierárquica com critério de decomposição baseado na planaridade, que posteriormente são utilizadas em um método de reconstrução baseado em funções de base radial. O método proposto tem como intuito principal eliminar limitações de métodos baseados na covariância de pontos em uma vizinhança Euclidiana. Foram efetuados testes sobre nuvens de pontos degradadas gradativamente.

**Palavras-chave**: Estimação de normais, nuvem de pontos, reconstrução de superfícies, função de base radial.

## Abstract

The generation of meshes from point clouds is a a problem frequently investigated in Computer Graphics and Geometric Modeling. The most typical point cloud reconstruction problem takes as an input a set o points with their associated normal vectors. One of the greatest challenges in point cloud reconstruction lies in estimating normals for arbitrary point clouds: clouds with numerous points, irregular sampling point clouds, noisy point clouds and point clouds with complex topology, for example, with tubular neighborhoods in different scales.

In this study, a new form of mesh reconstruction from point clouds is investigated. In this new method, normals are estimated using a hierarchical structure with a decomposing criterion based on planarity, which feeds a method of reconstruction based on radial basis functions. The main goal of the proposed method is to eliminate the limitations of methods based on covariance of points in a Euclidean neighborhood. Tests on gradually degraded point clouds were conducted.

Keywords: Normal estimation, point cloud, surface reconstruction, radial basis function.

# Lista de Figuras

3.1	Stanford Bunny, exemplo de nuvem de pontos digitalizada por um scanner laser	8
4.1	Exemplo de função de base radial. A influência do ponto de origem, no máximo da função, decai de forma radial	11
4.2	Exemplo subdivisão de um espaço bidimensional. Figura retirada de [22]	14
5.1	Exemplo de saida do Alpha Shapes com parâmetro $\alpha$ muito grande. Os pontos na superfície das orelhas do coelho são conectadas a pontos muito distantes.	21
6.1	Imagem original do <i>Stanford Bunny</i> , retirada de [2]	25
6.2	Imagem original do Happy Buddha, retirada de [2]	26
6.3	Imagem original do <i>Dragon</i> , retirada de [2]	26
6.4	Reconstrução com estimação de normais de Hoppe	27
6.5	Reconstrução com estimação de normais do algoritmo proposto	28
6.6	Histograma de erros entre a normal estimada e a original para <i>Stanford</i> <i>Bunny</i> com 0% de degradação	29
6.7	Histograma de erros entre a normal estimada e a original para <i>Stanford</i> <i>Bunny</i> com 10% de degradação	29
6.8	Histograma de erros entre a normal estimada e a original para <i>Stanford</i> <i>Bunny</i> com 20% de degradação	30
6.9	Histograma de erros entre a normal estimada e a original para $Stanford$ Bunny com 30% de degradação	30
6.10	Histograma de erros entre a normal estimada e a original para <i>Stanford</i> <i>Bunny</i> com 50% de degradação	31

6.11	Comparação entre estimações com 0% de degradação. Na esquerda o al- goritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	32
6.12	Comparação entre estimações com 10% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	32
6.13	Comparação entre estimações com 20% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	32
6.14	Comparação entre estimações com 30% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	33
6.15	Comparação entre estimações com 50% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	33
6.16	Comparação entre estimações, sem orientação, com 0% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido	33
6.17	Comparação entre estimações, sem orientação, com 10% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido	34
6.18	Comparação entre estimações, sem orientação, com 20% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido	34
6.19	Comparação entre estimações, sem orientação, com 30% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido	34
6.20	Comparação entre estimações, sem orientação, com 50% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido	35
6.21	Histograma de erros entre a normal estimada e a original para $Happy$ Buddha com 0% de degradação	36

6.22	Histograma de erros entre a normal estimada e a original para Happy Buddha com 10% de degradação	37
6.23	Histograma de erros entre a normal estimada e a original para <i>Happy</i> <i>Buddha</i> com 20% de degradação	37
6.24	Histograma de erros entre a normal estimada e a original para $Happy$ Buddha com 30% de degradação	38
6.25	Histograma de erros entre a normal estimada e a original para <i>Happy</i> <i>Buddha</i> com 50% de degradação	38
6.26	Comparação entre estimações com 0% de degradação. Na esquerda o al- goritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	39
6.27	Comparação entre estimações com 10% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	39
6.28	Comparação entre estimações com 20% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	39
6.29	Comparação entre estimações com 30% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	40
6.30	Comparação entre estimações com 50% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	40
6.31	Comparação entre estimações, sem orientação, com 0% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido	40
6.32	Comparação entre estimações, sem orientação, com 10% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido	41
6.33	Comparação entre estimações, sem orientação, com 20% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido	41

6.34	Comparação entre estimações, sem orientação, com 30% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido	41
6.35	Comparação entre estimações, sem orientação, com 50% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido	42
6.36	Histograma de erros entre a normal estimada e a original para <i>Dragon</i> com 0% de degradação	43
6.37	Histograma de erros entre a normal estimada e a original para <i>Dragon</i> com 10% de degradação	44
6.38	Histograma de erros entre a normal estimada e a original para <i>Dragon</i> com 20% de degradação	44
6.39	Histograma de erros entre a normal estimada e a original para <i>Dragon</i> com 30% de degradação	45
6.40	Histograma de erros entre a normal estimada e a original para <i>Dragon</i> com 50% de degradação	45
6.41	Comparação entre estimações com 0% de degradação. Na esquerda o al- goritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	46
6.42	Comparação entre estimações com 10% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	46
6.43	Comparação entre estimações com 20% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	46
6.44	Comparação entre estimações com 30% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	47
6.45	Comparação entre estimações com 50% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.	47

6.46	Comparação entre estimações, sem orientação, com $0\%$ de degradação. Na	
	esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e	
	na direita o algoritmo Híbrido	47
6.47	Comparação entre estimações, sem orientação, com $10\%$ de degradação.	
	Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto	
	e na direita o algoritmo Híbrido	48
6.48	Comparação entre estimações, sem orientação, com $20\%$ de degradação.	
	Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto	
	e na direita o algoritmo Híbrido	48
6.49	Comparação entre estimações, sem orientação, com $30\%$ de degradação.	
	Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto	
	e na direita o algoritmo Híbrido	48
6.50	Comparação entre estimações, sem orientação, com $50\%$ de degradação.	
	Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto	
	e na direita o algoritmo Híbrido	49

# Lista de Tabelas

6.1	Resultados do <i>Stanford Bunny</i>	28
6.2	Resultados Happy Buddha	36
6.3	Resultados Dragon	43

# Sumário

1	Intr	dução	1
	1.1	Motivação	1
	1.2	Objetivo	1
	1.3	Organização	2
2	Tral	alhos relacionados	3
3	Rec	nstrução de superfícies	5
	3.1	Introdução	5
	3.2	Problema	5
	3.3	Algoritmos	6
	3.4	Processo de reconstrução	7
		3.4.1 Digitalização da amostra	7
		3.4.2 Remoção de <i>outliers</i>	8
		3.4.3 Simplificação da amostra	8
		3.4.4 Remoção de ruído	9
		3.4.5 Estimação de normais	9
		3.4.6 Cálculo da superfície implícita	9
		3.4.7 Geração da malha	0
	3.5	Passos estudados 1	0
4	Fun	ões de base radial 1	1
	4.1	Introdução	1

	4.2	Algori	tmo RBF	12
	4.3	Algori	tmo RBF/POU	13
		4.3.1	Partição da unidade	13
		4.3.2	Subdivisão do espaço	14
		4.3.3	Funções de custo	15
5	Esti	mação (	de normais	16
	5.1	Introd	ução	16
	5.2	Proble	ema	16
	5.3	Algori	tmo de Hoppe	17
		5.3.1	Estimação das direções	17
		5.3.2	Cálculo de consistência dos sentidos das normais	18
	5.4	Algori	tmo proposto	20
		5.4.1	Estimação das direções	20
		5.4.2	Cálculo de consistência dos sentidos das normais	23
		5.4.3	Estimação de direções híbrida	24
6	Rest	ultados	computacionais	25
	6.1	Introd	ução	25
	6.2	Descri	ção dos Experimentos	26
		6.2.1	Simulação de Sombras	26
	6.3	Result	ados	27
		6.3.1	Resultados do <i>Stanford Bunny</i>	28
			6.3.1.1 Tabela de mínimos, máximos e médias	28
			6.3.1.2 Histogramas	28
			6.3.1.3 Estimações com orientação	32
			6.3.1.4 Estimações sem orientação	33

Re	eferênci	as			53
7	Conclu	usão e	trabalho	s futuros	51
	6.4 A	Análise	e dos Res	ultados	50
			6.3.3.4	Estimações sem orientação	47
			6.3.3.3	Estimações com orientação	46
			6.3.3.2	Histogramas	43
			6.3.3.1	Tabela de mínimos, máximos e médias	43
	6	5.3.3	Resultad	os do <i>Dragon</i>	43
			6.3.2.4	Estimações sem orientação	40
			6.3.2.3	Estimações com orientação	39
			6.3.2.2	Histogramas	36
			6.3.2.1	Tabela de mínimos, máximos e médias	36
	6	5.3.2	Resultad	os do Happy Buddha	36

## Capítulo 1

## Introdução

## 1.1 Motivação

Obter representações digitais precisas de objetos físicos ainda é um desafio para a área de computação gráfica. Existem diversas etapas entre a captura digital de um objeto e a visualização de um modelo digital do mesmo. Os algoritmos destas etapas são desenvolvidos na maioria dos casos em função do tipo de dado proveniente do dispositivo de captura.

As técnicas de captura, como as baseadas em scanners 3D ou tomografias, retornam uma discretização digital do objeto que raramente é precisa. Ruído, *outliers*, dados faltando e outros problemas de captura estão sempre presente nas saídas de dados dos dispositivos atuais. Por isso, os algoritmos de reconstrução precisam ser robustos o suficiente para tratar dados com diferentes propriedades. Esta necessidade de robustez motivou a investigação apresentada neste trabalho, que combina dois algoritmos conhecidos da área, o algoritmo de estimação de normais de Hoppe [12] e o algoritmo de cálculo de superfície implícita por funções de base radial [7].

### 1.2 Objetivo

Este trabalho propõe uma alternativa à estimação de normais comumente referenciada na literatura: algoritmo apresentado por Hugues Hoppe [12]. O algoritmo de Hoppe requer uma nuvem de pontos com densidade tal que a distância máxima entre dois pontos da nuvem seja menor que o menor tamanho de característica da superfície representada (*local feature size*, definida em [11]).

O método proposto por Hoppe, para estimar normais, se baseia na construção de um grafo de Riemmann dado por uma árvore geradora mínima, acrescida de arestas em uma vizinhança determinada por uma distância Euclidiana. No método proposto por Hoppe, em casos onde a nuvem é formada por uma amostragem não uniforme, o uso de distância euclidiana para construção do grafo de Riemmann pode trazer dificuldades. Um exemplo é o caso no qual uma vizinhança de pontos advém de uma vizinhança tubular com uma dada amostragem e uma densidade diferente fora dessa região. Nesse caso o algoritmo tende a agrupar pontos que estão em lados distintos da superfície.

O objetivo do trabalho aqui proposto é desenvolver um algoritmo para reconstrução de nuvens de pontos baseados em funções de base radial, que seja capaz de lidar com distribuições de pontos muito densas, mas que ao mesmo tempo possuam uma amostragem não uniforme. A amostragem é não uniforme quando diferentes partes de um mesmo objeto possuem densidades diferentes ou quando existem buracos causados por sombras na captura.

### 1.3 Organização

O capítulo 2 apresenta o problema de estimação de normais, no contexto da reconstrução de superfícies, e trabalhos relacionados. No capítulo 3, está descrito o processo de reconstrução de superfícies abordado por este trabalho. O capítulo 4 descreve os fundamentos de funções de base radial e sua aplicação na área de reconstrução de superfícies. A estimação de normais é descrita no capítulo 5, que abrange tanto o algoritmo clássico de estimação de normais quanto o algoritmo proposto deste trabalho. O capítulo 6 contém os resultados obtidos e discussão dos mesmos. No capítulo 7 é apresentada a conclusão e trabalhos futuros.

## Capítulo 2

## Trabalhos relacionados

Uma nuvens de pontos é uma representação digital discreta da superfície de um objeto analógico, obtida através de um scanner 3D. Em alguns casos, em particular quando a densidade é baixa, este tipo de dado não é satisfatório para manipulação e visualização em um ambiente digital. Para tais tarefas é mais adequada uma superfície implícita ou uma malha de polígonos.

A reconstrução de superfícies a partir de nuvens de pontos estuda esta conversão de dados: de uma representação discreta de pontos para uma superfície implícita ou malha de polígonos. É importante notar que a qualidade da representação gerada é dependente da qualidade da entrada de dados. Os algoritmos existentes, embora robustos, ainda são sensíveis aos diversos fatores que afetam uma amostra, como, por exemplo ruído, *outliers*, dados faltando e densidade variável. Portanto, ainda há espaço para estudo de melhorias em relação a estes fatores.

No estado da arte da área de reconstrução de superfícies [5] são apresentados diversos algoritmos de reconstrução; alguns dos algoritmos mais famosos requerem normais dos planos tangentes da superfície associadas aos pontos. A necessidade de uma normal de um plano tangente existe porque, no contexto destes algoritmos, é possível reconstruir uma superfície implícita contínua de infinitas formas, apenas com os pontos das nuvens como dado de entrada. Além disso, alguns algoritmos precisam diferenciar a parte interna da parte externa do objeto.

A seção 3.4 apresenta o processo de reconstrução de superfícies utilizado. Este processo pode ser resumido nos passos a seguir:

1. Digitalização da amostra: um scanner digitaliza um objeto analógico em uma amostragem discreta de pontos num espaço 3D.

- Simplificação e limpeza da amostra: passos de simplificação, remoção de ruído e *outliers* para ordenar a amostra de acordo com os parâmetros de entrada do algoritmo de reconstrução.
- 3. Estimação de normais: este é o passo estudado neste trabalho. A estimação de normais calcula planos tangentes à superfície em cada ponto da amostra. O estudo deste passo é apresentado no capítulo 5.
- Cálculo da superfície implícita: neste trabalho, a superfície implícita é calculada através de funções de base radial (RBFs) [7], que necessitam dos pontos e de normais orientadas associadas.
- 5. Geração da malha: caso seja necessária uma malha de polígonos, existem algoritmos para extração de malhas poligonais a partir de uma superfície implícita.

A estimação de normais comumente utilizada é apresentada por Hoppe em [12]. Esta estimação de normais se baseia no cálculo de planos tangentes locais através da análise da vizinhança Euclidiana em cada ponto. Este trabalho foca no estudo deste algoritmo e a busca por uma melhoria.

Outros trabalhos de detecção de planos e geometria em nuvens de pontos podem ser destacados. A Transformada de Hough pode ser utilizada para detectar planos, preservando características não suaves do modelo em [6] e [17]. Em [9] é estudada uma estimação de normais baseada em diagrama de Voronoi.

O trabalho em [18] detecta características geométricas baseado no método de análise de componentes principais [13], também utilizada por Hoppe, introduzindo covariância ponderada por uma média geométrica. Os pontos possuem um atributo de peso para demonstrar sua contribuição espacial. Outros métodos estatísticos são utilizados para detectar o melhor plano tangente local em [16].

## Capítulo 3

## Reconstrução de superfícies

### 3.1 Introdução

Uma reconstrução de superfícies consiste em recuperar a representação digital de um objeto físico amostrado. Esta amostra do objeto, que é o dado de entrada de um algoritmo de reconstrução, é obtida através de um dispositivo de captura.

Um método utilizado para a reconstrução de uma superfície tem uma forte conexão com a técnica de captura utilizada. Existem diversos métodos de reconstrução de superfícies. Cada método lida com um tipo específico de amostra, proveniente de uma das diversas técnicas de captura. Com a diversidade de dispositivos aumentando e os dados de capturas cada vez mais acessíveis, se faz necessário um contínuo estudo e aperfeiçoamento dos métodos existentes de reconstrução, assim como a criação de novos métodos aptos às novas técnicas de captura.

Este trabalho estuda um grupo específico de algoritmos de reconstrução de superfícies, seguindo o processo descrito na seção 3.4. A seção 3.2 descreve o problema abordado de reconstrução. Os algoritmos são especificados na seção 3.3.

### 3.2 Problema

O problema de reconstrução de superfícies pode ser formulado como: dada uma nuvem de pontos no espaço tridimensional, que é uma amostra digital de um objeto físico, encontrar uma superfície que seja uma aproximação da superfície que descreve o objeto.

Esta nuvem pode ter propriedades diferentes de acordo com o formato da captura. Os algoritmos existentes obtém bons resultados para nuvens bem comportadas, sem ruído,

sem dados faltando e com a entrada de dados requisitada pelo algoritmo. No entanto, este tipo de dado é irreal para as tecnologias de captura atuais. Por isso a necessidade de se estudar melhorias que tornem os métodos mais robustos, capazes de encontrar soluções para nuvens cujas propriedades não respeitam as pré-condições assumidas pelos algoritmos analisados.

Este trabalho investiga a combinação da reconstrução por funções de base radial [7], um algoritmo robusto a falhas na amostra e ruído, com a estimação de normais de Hoppe [12], um algoritmo sensível a falhas na amostra, que é capaz de estimar as normais necessárias para a reconstrução por funções de base radial.

### 3.3 Algoritmos

No estudo de Berger et al. sobre o estado da arte da área de reconstrução de superfícies [5], os algoritmos mais conhecidos da área são relacionados de acordo com sua robustez, requisitos de entrada de dados, classe dos objetos amostrados e saída de dados da reconstrução.

Este trabalho se concentra nos algoritmos que retornam superfícies implícitas como saída de dados, mais precisamente aqueles que requerem normais orientadas associadas aos dados de entrada. Os exemplos mais comuns deste tipo de algoritmo são aqueles baseados em funções de base radial [7], ou *radial basis functions* (RBFs), reconstrução de Poisson [15] e partição da unidade [21]. Os algoritmos baseados em RBFs foram selecionados como foco para este trabalho.

Embora os algoritmos baseados em RBFs necessitem de normais orientadas, é possível que uma entrada de dados originalmente sem normais ainda possa ser útil para tais algoritmos de reconstrução. A chave para este processo é um outro algoritmo de reconstrução de superfícies apresentado no estudo de Berger: o algoritmo baseado em planos tangentes de Hoppe et al. [12].

O algoritmo de Hoppe não necessita de normais orientadas, pois inicialmente gera uma aproximação de uma superfície preliminar através da estimação de planos tangentes em cada pontos da amostra. Estes planos tangentes são representados por normais, ou seja, o passo inicial apresentado por Hoppe nada mais é do que uma estimação das normais orientadas.

Este trabalho é baseado em uma modificação do processo de estimação de normais

apresentado por Hoppe para criar uma amostra com normais orientadas, a partir da amostra original, para os algoritmos de reconstrução por RBFs. Na CGAL [1], biblioteca amplamente utilizada em geometria computacional, o algoritmo de Hoppe é a única referência para a estimação de normais nas amostras analisadas [3].

É importante destacar que o algoritmo de Hoppe possui limitações, quanto a falhas na amostra, como dados faltando, ruído, amostragem irregular e *outliers*. Estas, por sua vez, limitam todo o processo de reconstrução apresentado. Este trabalho aborda este problema e apresenta uma proposta para mitigar seu impacto na reconstrução, aumentando a robustez do algoritmo.

### 3.4 Processo de reconstrução

O processo da reconstrução estudado neste trabalho é baseado em uma sequencia de etapas [4], descritas a seguir: Digitalização da amostra, que produz um conjunto de pontos; Remoção de *outliers*; Simplificação da amostra, para diminuir o número de pontos no conjunto de entrada; Remoção de ruído; Estimação de normais; Cálculo da superfície implícita; Geração da malha.

Embora todo o processo esteja descrito, este trabalho se concentra nos passos de estimação de normais e cálculo de superfície implícita. As outras etapas são descritas nesta seção apenas para uma visão geral do processo.

### 3.4.1 Digitalização da amostra

O objeto é digitalizado por um dispositivo que retorna um conjunto de pontos, ou amostra, discreto no espaço tridimensional. A este conjunto dá-se o nome de nuvem de pontos (*point cloud*).

Idealmente, uma nuvem de pontos é uma amostra da superfície do objeto físico. No entanto, é comum que uma captura contenha imprecisões, que podem causar ruído; falhas na representação, como buracos ou desalinhamento; outliers; ou densidade irregular.

Embora este trabalho apresente algoritmos cuja entrada tradicionalmente é criada por scanners laser de objetos, nenhuma técnica de captura é estudada. Assume-se que a nuvem de pontos possui determinadas propriedades de acordo com a necessidade de cada algoritmo de reconstrução analisado.



Figura 3.1: Stanford Bunny, exemplo de nuvem de pontos digitalizada por um scanner laser

### 3.4.2 Remoção de outliers

*Outliers* são definidos como dados de uma amostra que estão muito fora da variação esperada de um determinado experimento. Em uma nuvem de pontos, um *outlier* pode ser identificado como um ponto que está distante da maior parte dos outros pontos. *Outliers* são considerados erros na captura e devem ser removidos da amostra.

O procedimento mais comum para tratar *outliers* é remover os pontos da amostra com maior distância média para seus k vizinhos. Os parâmetros deste algoritmo são o valor k, que indica o número de pontos nas vizinhanças analisadas, e o percentual de pontos da amostra total que deve ser retirado. Este trabalho considera que a nuvem de pontos não possui *outliers*.

### 3.4.3 Simplificação da amostra

Quando a quando a nuvem de pontos possui uma densidade desnecessariamente grande, é possível simplificá-la para diminuir a carga de processamento e memória nos próximos passos do processo. Um exemplo desta etapa é a simplificação aleatória, na qual pontos da amostra são retirados aleatoriamente até um determinado parâmetro de percentual de remoção. No entanto, uma simplificação grosseira, como a aleatória, pode causar buracos na nuvem e, consequentemente, distorções na superfície reconstruída. Então é importante que os dados de entrada sejam avaliados antes desta etapa, buscando viabilizar uma simplificação mais adequada.

Uma função de simplificação, apresentada em [3], divide o espaço em um *grid*, agrupando os pontos que estão contidos em uma mesma célula. A simplificação é feita elegendo um membro arbitrário de cada célula para representar o agrupamento.

#### 3.4.4 Remoção de ruído

A digitalização de um objeto físico pode gerar ruídos na amostra. Estes ruídos compõem um dos desafios da área de reconstrução. Um dado demasiadamente ruidoso não pode ser reconstruído com precisão, pois o próprio dado de entrada não representa mais uma amostra da superfície do objeto físico.

Um exemplo de algoritmo de remoção de ruído [3] projeta pontos em uma superfície aproximada na vizinhança dos pontos da nuvem. Esta projeção aproximada serve para suavizar a variação dos pontos da entrada, como o ato de borrar as cores de uma imagem bidimensional. Este trabalho considera que a nuvem de pontos possui pouco ruído.

#### 3.4.5 Estimação de normais

Uma normal orientada associada a um ponto é a representação da tangente da superfície do objeto físico amostrado. A orientação da normal indica qual parte é externa e qual parte é interna à superfície. Os algoritmos de cálculo de superfície implícita deste trabalho requerem que os pontos da nuvem tenham normais orientadas associadas. Embora algumas técnicas de captura já retornem este tipo de dado, este trabalho considera que as normais ainda não foram calculadas.

### 3.4.6 Cálculo da superfície implícita

Neste passo do processo, considera-se que os dados discretos da nuvem pertencem ao espaço definido por um sistema de funções implícitas que representa uma aproximação da superfície.

Para representar a superfície através de funções implícitas é possível utilizar funções de base radial (RBFs) [7], que são funções no domínio dos números reais cujos valores dependem da distância a partir de um determinado ponto. No caso, os pontos do conjunto de entrada.

Sabe-se da literatura que não é possível simplesmente aproximar uma nuvem de pontos por uma superfície implícita determinada por RBFs caso apenas os pontos da fronteira do objeto sejam conhecidos. Isso decorre do fato que o sistema pode assumir uma solução trivial nula que não é de interesse para a reconstrução.

Para obter uma solução não trivial do sistema, pontos fora e dentro da fronteira são criados e adicionados ao domínio das funções. Estes pontos são criados na direção das normais de cada ponto da superfície.

### 3.4.7 Geração da malha

Como a superfície implícita encontrada é uma função matemática, se faz necessário algum método gráfico para visualização do resultado. Um método simples de visualização é gerar uma malha poligonal, de triângulos, que é a representação gráfica da superfície matemática.

Dentre os diferentes algoritmos para extração de malhas poligonais de dados implícitos pode-se destacar o Marching Cubes [19] e o Dual Contouring [14]. Este trabalho optou por utilizar o primeiro. Considerando que a entrada do algoritmo de geração de malha é a superfície implícita, como uma função no espaço, o Marching Cubes consiste em dividir o espaço em um reticulado, de modo que os vértices dos cubos deste reticulado tenham como atributo seus valores na função de entrada. De acordo com uma tabela pré-determinada pelo Marching Cubes, combinações de valores positivos e negativos nos vértices de um cubo retornam polígonos que formam a malha.

### 3.5 Passos estudados

Este trabalho é um estudo da estimação de normais. Como a entrada do passo de cálculo de superfície implícita é a saída da estimação de normais, é importante compreender o contexto deste cálculo. Este contexto é apresentado neste trabalho como um estudo das funções de base radial para cálculo da superfície implícita.

## Capítulo 4

## Funções de base radial

## 4.1 Introdução

Uma função de base radial, ou *radial basis function* (RBF), é uma função cujos valores dependem apenas da distância a partir de uma origem. Funções de distância, como as RBFs, podem ser utilizadas na reconstrução de uma superfície representada por uma nuvem de pontos, considerando estes pontos como as origens. A superfície reconstruída é definida implicitamente pelo conjunto zero da função de distância. Uma função implícita que aproxima a superfície do objeto físico capturado pode ser calculada pelo algoritmo de planos tangentes de Hoppe et al. [12]. Na figura 4.1 pode ser vista uma gaussiana, um tipo de função de base radial. Com um ponto de origem no máximo da função, ela representa a influência que este ponto exerce ao seu redor.



Figura 4.1: Exemplo de função de base radial. A influência do ponto de origem, no máximo da função, decai de forma radial.

Uma característica desejável do algoritmo que calcula a superfície implícita, é que apresente robustez quanto a presença de poucas normais com direções ou sentidos incorretos no conjunto de entrada. A função de distância definida por Hoppe não apresenta essa característica, porque, como cada plano tangente é a base do cálculo da função distância nos pontos mais próximos a ele, uma normal do plano tangente com orientação incorreta gera uma região do espaço em que a função de distância não representa a superfície. Por isso a escolha de utilizar métodos alternativos.

### 4.2 Algoritmo RBF

O método utilizado, descrito em [23], consiste em definir uma função de distância e ajustar RBFs a esta função. Como apresentado por Hoppe [12], a função de distância é definida como  $f : \mathbb{R}^3 \to \mathbb{R}$ , de forma que f estima a distância à superfície ainda desconhecida. Portanto, dados os pares de pontos  $p_i$  e valores  $h_i$ , é necessário encontrar uma função ftal que  $f(p_i) = h_i$ .

Uma RBF é uma função qualquer da forma  $\phi(p - x_j)$ , baseada na distância entre um ponto p e uma origem  $x_j$  pertencente à amostra. Logo, a equação de reconstrução da superfície é dada por:

$$f(p) = \sum \omega_j \phi(p - x_j) + \pi(p) \tag{4.1}$$

Onde  $\omega_j$  é o peso de uma origem  $x_j, \phi : \mathbb{R}^3 \to \mathbb{R}$  é uma função de base radial e  $\pi(p)$ é um polinômio dependente da escolha de  $\phi$ . Portanto, substituindo  $f(p_i)$  por  $h_i$ :

$$h_i = \sum \omega_j \phi(p_i - x_j) + \pi(p_i) \tag{4.2}$$

Esta equação determina um sistema linear positivo e semi-definido, cujo vetor solução contém os pesos  $\omega_j$  e os valores do polinomio  $\pi$ . Com esses valores, a função festá determinada, pronta para ser avaliada em qualquer ponto do espaço, definindo uma reconstrução da superfície.

O sistema de equações tem uma solução trivial nula caso apenas os pontos originais sejam utilizados. Isso ocorre porque  $h_i = 0$  para qualquer ponto da entrada, pois é considerado que todos os pontos da entrada pertencem a superfície.

Para evitar a solução nula, são criados pontos extras, fora da superfície, cujo valor  $h_i$  é, convencionalmente, definido como +1 para pontos na área externa ao objeto e -1 na área interna ao objeto. Para criar estes pontos, basta utilizar a direção e sentido das

normais encontradas no capítulo 5. Como notado em [7], é importante que a criação de pontos tenha a condição de que o ponto original mais próximo ao ponto criado seja aquele que tem a normal associada que foi utilizada para a criação.

## 4.3 Algoritmo RBF/POU

O sistema de equações apresentado na seção 4.2 é inviável para uma grande entrada de dados, pois os recursos de armazenamento são escassos em comparação a um número tão grande de variáveis. Para mitigar este problema, este trabalho utiliza o método apresentado em [22], que utiliza uma partição de unidades com as funções de base radial, descrita nessa seção. Embora outras soluções já foram apresentadas na literatura, como uma redução da utilização dos pontos apresentada em [7], esta solução de partição de unidade foi escolhida por sua simplicidade. É possível que uma combinação de técnicas seja viável, porém, desnecessária para as entradas de dados analisadas.

O método de partição de unidade com funções de base radial subdivide o espaço, associando as subdivisões a uma família de funções que formam uma partição da unidade. Dentro de cada subdivisão do espaço é utilizado o método de RBF descrito, que gera soluções locais de aproximação da superfície. Estas soluções locais são combinadas através da partição da unidade para gerar uma função contínua.

#### 4.3.1 Partição da unidade

Um domínio global  $\Omega$  é dividido em M subdomínios ligeiramente sobrepostos  $\Omega_i$ , com  $\Omega \subseteq \bigcup_i \Omega_i$ . Nesses subdomínios é construída uma partição da unidade, que é um conjunto de funções não negativas  $w_i$  com suporte  $supp(w_i) \subseteq \Omega_i$  e  $\sum w_i = 1$  em todo  $\Omega$ .

Cada subdomínio  $\Omega_i$  define um subconjunto de pontos da nuvem através de uma subdivisão do espaço. Esse subconjunto é utilizado como entrada para calcular uma reconstrução local  $f_i$ , utilizando o método descrito na seção 4.2. A solução global, então, é uma combinação das funções locais com os pesos das partições:

$$F(p) = \sum f_i(p)w_i(p) \tag{4.3}$$

A condição  $\sum w_i = 1$  é dada normalizando um conjunto de funções  $W_i$  contínuas nas regiões de sobreposição de  $\Omega_i$ :



Figura 4.2: Exemplo subdivisão de um espaço bidimensional. Figura retirada de [22].

$$w_i(p) = \frac{W_i(p)}{\sum_j W_j(p)} \tag{4.4}$$

#### 4.3.2 Subdivisão do espaço

Considerando um conjunto X de pontos contidos em  $\Omega_i$ , o domínio  $\Omega_i$  é subdividido caso o número de pontos n em X seja maior do que um parâmetro que especifica um número máximo de pontos  $T_{max}$  em um domínio. A subdivisão ocorre recursivamente até que a condição  $n < T_{max}$  seja satisfeita, então o subdomínio com esta condição é incluído no conjunto da partição da unidade. Para garantir as áreas de sobreposição, importantes para a continuidade da função global, as subpartições são criadas ligeiramente dilatadas.

Uma partição que contém poucos pontos pode criar resultados inesperados para  $f_i$ , por isso existe um parâmetro de número mínimo de pontos  $T_{min}$ . No caso de  $n < T_{min}$ , o tamanho de uma partição é aumentado. Nota-se que ao aumentar uma partição, é possível que n se torne maior que  $T_{max}$ , logo este ponto do algoritmo faz um ajuste fino, com funções para aumentar e diminuir uma partição, para que todas as partições possam respeitar as condições dos parametros  $T_{min}$  e  $T_{max}$ .

Neste trabalho, os domínios  $\Omega_i$  são paralelepipedos alinhados aos eixos. Definidos por dois pontos opostos  $S \in T$ , com S mais próximo da origem. Em um espaço 3D, um domínio  $\Omega_i$  é subdividido em oito e, considerando que d é o fator de sobreposição, cada subdomínio é criado com base nas fórmulas:

$$S' = S - \frac{(d-1)(T-S)}{4}$$
e  
$$T' = \frac{S+T}{2} + \frac{(d-1)(T-S)}{4}$$
A função que muda o tamanha

A função que muda o tamanho de uma partição segue as fórmulas: (k-1)(T-S)

$$S' = S - \frac{(k-1)(T-S)}{2}$$
  
e  
$$T' = T + \frac{(k-1)(T-S)}{2}$$

Onde k é o fator de dilatação ou redução.

### 4.3.3 Funções de custo

A função de custo  $W_i$  determina a continuidade da função global F nas regiões de sobreposição dos subdomínios  $\Omega_i$ . A função de custo definida por [22] é a composição de uma função de distância  $D_i : \mathbb{R}^n \to [0, 1]$  e uma função de decaimento  $V : [0, 1] \to [0, 1]$ , com  $D_i(p) = 1$  nas fronteiras de  $\Omega_i$  e  $W_i(p) = V \circ D_i(p)$ . A função de distância utilizada é:

$$D_i(p) = 1 - \prod_{r \in x, y, z} \frac{4(p^{(r)} - S^{(r)})(T^{(r)} - p^{(r)})}{(T^{(r)} - S^{(r)})^2}$$
(4.5)

A função de decaimento utilizada é:

$$V(d) = 1 - d \tag{4.6}$$

## Capítulo 5

## Estimação de normais

## 5.1 Introdução

Neste capítulo estudamos a estimação de um plano tangente, representado por um vetor normal, associado a um ponto, que aproxima a superfície do objeto original no espaço próximo àquele ponto. A estimação de planos tangentes, conhecida também como estimação de normais, é uma etapa fundamental para a reconstrução de superfícies. O algoritmo mais conhecido foi criado por Hoppe [11].

Um conjunto de normais bem estimado é importante para que o resultado da reconstrução seja razoável. O algoritmo de Hoppe obtém bons resultados quando a densidade da nuvem é regular, com amostragem uniforme, e quando a menor característica do objeto representado é maior que a distância entre quaisquer dois pontos da nuvem. Este trabalho propõe e investiga uma alternativa ao algoritmo de Hoppe, que seja capaz de atingir resultados robustos sem assumir essas propriedades.

Este capítulo descreve o problema de estimações de normais e os algoritmos propostos para sua solução. A seção 5.2 apresenta o problema de estimação de normais. Na seção 5.3 é descrito o algoritmo de Hoppe. O algoritmo proposto por este trabalho para estimação de normais é apresentado na seção 5.4.

### 5.2 Problema

Dado um ponto  $x_i$  da nuvem, determinar o plano tangente  $Tp(x_i) = (o_i, n_i)$  que aproxima a superfície original na região próxima a  $x_i$ , com  $o_i$  representando o centro do plano e  $n_i$ representando seu vetor normal.

### 5.3 Algoritmo de Hoppe

O algoritmo de Hoppe consiste em definir uma função  $f: D \to \mathbb{R}$ , onde  $D \subset \mathbb{R}^3$  é uma região na vizinhança dos dados, de forma que f estima a distância à superfície ainda desconhecida. Em um modelo implícito, o conjunto dos pontos  $x \in D$  tais que f(x) = 0, chamado zero da função Z(f), é a estimativa da superfície. Para definir essa função de distância, um plano tangente é associado para cada ponto da nuvem. Pode-se considerar que estes planos são, localmente, aproximações lineares da superfície.

Nota-se que o algoritmo baseado em planos tangentes é, na verdade, uma solução para o problema de reconstrução de superfícies: a função f define implicitamente a superfície de forma local. No entanto, a função definida é pouco robusta a erros no cálculo dos planos tangentes. Por isso, a criação de planos tangentes pode ser vista como uma estimação de normais utilizada para computar uma outra função implícita.

O algoritmo de Hoppe para estimação de normais é dividido em duas partes. A primeira estima a direção dos vetores normais do plano tangente aproximado em cada ponto da nuvem, e a segunda parte corrige o sentido das mesmas para que estejam globalmente consistentes, de acordo com a convenção escolhida: todas para fora ou todas para dentro do modelo.

#### 5.3.1 Estimação das direções

Para determinar o centro e a normal do plano  $Tp(x_i)$  é necessária uma vizinhança definida pelos k pontos mais próximos a  $x_i$ , denotada  $Nbhd(x_i)$ . O ponto  $o_i$  é o centróide desta vizinhança, calculado como:

$$o_i = \frac{\sum_{i \in Nbhd(x_i)} x_i}{k} \tag{5.1}$$

A função de distância de um ponto arbitrário p para  $Tp(x_i)$  é definida como:

$$d_i(p) = (p - o_i).n_i$$
(5.2)

O centro e a normal são calculados de modo que o plano  $d_i(p) = 0$  é o melhor ajuste de mínimos quadrados para a vizinhança  $Nbhd(x_i)$ .

A normal  $n_i$  é calculada utilizando analise de componentes principais [13], ou principal

*component analysis* (PCA). Este cálculo é uma aproximação, a partir dos dados obtidos, de um elipsóide n-dimensional cujos eixos são as componentes principais. Esses eixos podem ser encontrados através dos autovetores de uma matriz de covariância dos dados. Cada autovetor, ortogonal e unitário, pode ser interpretado como um eixo do elipsóide.

O plano tangente é um plano formado pelos dois vetores representantes dos maiores eixos do elipsóide e a normal  $n_i$  é o vetor perpendicular a esses dois vetores. Por isso, pode-se considerar que a normal  $n_i$  é o autovetor que representa o menor eixo. Primeiro é criada uma matriz de covariância de  $Nbhd(x_i)$ , que é a matriz  $3 \times 3$  positiva semi-definida

$$CV = \sum_{p \in Nbhd(x_i)} (p - o_i) \otimes (p - o_i)$$
(5.3)

onde  $\otimes$  é o produto tensorial.

Se  $\lambda_1 \ge \lambda_2 \ge \lambda_3$  são os autovalores de CV associados aos autovetores  $v_1$ ,  $v_2$  e  $v_3$ , podemos definir que  $n_i$  é  $v_3$  ou  $-v_3$ . O sentido escolhido determina a orientação do plano tangente e será tratado no segundo passo do algoritmo. A relação entre os três autovalores é relevante para uma boa estimação do plano. É importante notar que o elipsóide pode se deformar em uma tendência esférica quanto menor for a diferença entre os autovalores. Como a esfera não tem componentes diferentes entre si, a estimação do plano tangente não é confiável.

O algoritmo do Hoppe assume que a entrada de dados é uma nuvem com densidade regular cuja menor característica do objeto representado é maior do que a distância entre dois pontos da nuvem.

### 5.3.2 Cálculo de consistência dos sentidos das normais

Para que a superfície seja aproximada de forma correta, planos tangentes de pontos geometricamente próximos devem ter o sentido de suas normais consistentemente apontando para o mesmo lado. Embora os algoritmos de cálculo da superfície implícita sejam robustos a algumas normais da amostra com direção incorreta, a orientação inconsistente entre um número razoável de planos tangentes é suficiente para inviabilizar uma reconstrução.

Assumindo que a superfície é suave e a nuvem é densa o suficiente, pode-se supor que, caso a distância geométrica entre dois pontos  $x_i$  e  $x_j$  seja pequena, seus planos tangentes associados,  $Tp(x_i) = (o_i, n_i)$  e  $Tp(x_j) = (o_j, n_j)$ , são quase paralelos, ou seja  $n_i \cdot n_j$  é aproximadamente +1 ou -1. O valor positivo identifica normais com sentidos consistentes entre si, enquanto o valor negativo identifica que uma das duas normais precisa ser invertida. Para dois pontos isolados a ideia é simples, no entanto, calcular a consistência global da orientação das normais estimadas pelo algoritmo se mostra um desafio maior do que a estimação das direções das mesmas.

Este problema pode ser modelado como um problema de otimização em um grafo. Suponha o grafo não direcionado G = (V, E) cujo conjunto de vértices V está associado aos centros  $o_i$  dos planos  $Tp(x_i)$  para todo  $x_i$ , e o conjunto E contem arestas  $(o_i, o_j)$  tal que os centros  $o_i$  e  $o_j$  estejam próximos. Esta proximidade pode ser determinada de duas formas: ou os dois pontos são encontrados na vizinhança um do outro mutuamente; ou a distância entre eles é menor que um parâmetro pré-determinado, relacionado à densidade da nuvem. O grafo G, chamado de Grafo de Riemann ou Grafo Riemanniano, codifica a proximidade geométrica entre planos tangentes.

Na otimização deste grafo, o custo w(e) de uma aresta  $e = (o_i, o_j)$  é o grau em que  $Tp(x_i)$  e  $Tp(x_j)$  têm orientações consistentes, com  $w(e) = n_i \cdot n_j$ . O resultado da otimização, então, é a escolha de pesos  $b_i$  pertencentes a [-1, 1], que definem a orientação final  $b_i n_i$ , tal que seja maximizado o custo dado por:

$$\sum_{(i,j)\in E} b_i b_j w(i,j) \tag{5.4}$$

Este problema é demonstrado como um problema NP-Difícil em [11]. Portanto, um algoritmo exato possui uma complexidade inviável para dados de entrada muito grandes.

Uma solução aproximada consiste em propagar, de forma gulosa, as orientações dos planos entre os vértices conectados no grafo G. O algoritmo escolhe um vértice arbitrário para começar e propaga sua orientação pelos seus vizinhos de forma que  $w(e) = n_i . n_j$ seja positivo. Caso w(e) seja negativo, a normal  $n_j$  é invertida. O algoritmo segue sua propagação nos vizinhos até que todos os vértices sejam visitados.

Uma escolha previsível e conveniente do vértice inicial é começar por aquele associado ao plano com centro de maior valor possível na coordenada z e fazer com que a orientação desse plano aponte no sentido do eixo +z. Isso garante que ao menos o vértice inicial tem o plano tangente orientado para fora do modelo.

Por ser uma solução aproximada e a escolha do vértice inicial ser arbitrária, a ordem de propagação da orientação entre planos é relevante para o resultado do algoritmo. A propagação gulosa pode ser melhorada caso ela favoreça planos próximos com direções muito parecidas, independente da orientação. Para isso, é associado o custo  $1 - |n_i \cdot n_j|$ para cada aresta (i, j) do grafo G. Além de ser não-negativo, o custo é menor quanto mais perto de paralelas são as direções do planos tangentes.

A árvore geradora mínima do grafo G, com custos associados, é criada. Esta árvore relaciona os planos geometricamente próximos com direções mais parecidas. Usando como raiz o vértice inicial descrito previamente, as orientações são propagadas em uma travessia em profundidade na árvore. O algoritmo termina com uma aproximação da superfície original representada por seus dados de saída: uma nuvem de pontos com normais associadas, representando o plano tangente que aproxima a superfície em cada ponto.

## 5.4 Algoritmo proposto

A diferença fundamental entre a proposta deste trabalho e o algoritmo de Hoppe está na estimação dos planos tangentes, realizada no primeiro passo do algoritmo. Ao invés de aproximar localmente planos tangentes em cada ponto, o algoritmo proposto agrupa pontos aproximadamente coplanares e calcula os planos tangentes aproximados de acordo com este agrupamento. O passo de cálculo de consistência do sentido das normais não foi modificado.

#### 5.4.1 Estimação das direções

O algoritmo do Hoppe obtém bons resultados quando a entrada de dados é uma nuvem com densidade regular cuja menor característica do objeto representado é maior do que a distância entre dois pontos da nuvem. A mudança na estimação de normais proposta nesta seção visa lidar com conjuntos de entrada que não satisfaçam estas condições de regularidade e uniformidade. Neste caso, dado um ponto  $x_i$  e os pontos pertencentes à sua vizinhança  $Nbhd(x_i)$ , não é possível afirmar que todos representam a mesma parte do objeto, ou seja, um plano tangente formado por este conjunto não necessariamente aproxima a superfície original.

Esta vizinhança mal formada sugere que é necessário obter mais informações da nuvem original antes de calcular os planos tangentes. Idealmente, um ponto p pertencente a uma vizinhança geométrica  $Nbhd(x_i)$  deveria pertencer à vizinhança geodésica de  $x_i$ , ou seja, para que o ponto p seja elegível a participar do cálculo do plano tangente associado a  $x_i$ , ele necessariamente deve estar próximo a  $x_i$  na superfície. Como o objetivo do algoritmo é encontrar a superfície aproximada, obviamente a informação das geodésicas da mesma não faz parte da entrada de dados. Resta encontrar alguma relação entre os pontos da nuvem da qual se possa extrair algum tipo de informação de uma aproximação grosseira da superfície.

Antes da elaboração da proposta apresentada nesta seção, estratégias alternativas para obter informações das geodésicas foram testadas. Uma ideia inicial se baseou no algoritmo Alpha Shapes [10], que obtém a triangulação da forma representada pelos pontos da nuvem de entrada. É intuitivo pensar que um algoritmo que consiga aproximar a forma de um objeto, como o Alpha Shapes, seja suficiente para a tarefa de encontrar as relações das geodésicas da superfície.

O algoritmo Alpha Shapes é baseado na calibração de um parâmetro  $\alpha$ , que determina a distância máxima permitida para que dois pontos do conjunto de entrada sejam considerados vizinhos. No entanto, o Alpha Shapes se mostrou incapaz de lidar com a variação da densidade da amostra: com o parâmetro  $\alpha$  grande demais a aproximação é quase um fecho convexo; com o parâmetro pequeno demais, poucas relações entre pontos são formadas, gerando informações insuficientes.



Figura 5.1: Exemplo de saida do Alpha Shapes com parâmetro  $\alpha$  muito grande. Os pontos na superfície das orelhas do coelho são conectadas a pontos muito distantes.

Para esta entrada de dados com densidade irregular, o Alpha Shapes gera um modelo grosseiro demais para ser confiável, embora crie informações relevantes para o cálculo dos planos tangentes. Por isso, esta opção não é viável. A alternativa proposta neste trabalho consiste em subdividir o espaço em função da análise das componentes principais dos pontos, de modo que cada sub-região agrupe pontos que representem a aproximação de um plano. Idealmente, esta divisão do espaço seria capaz de agrupar, nas sub-regiões, pontos que representam a mesma região da superfície a ser aproximada.

Esta ideia é baseada na detecção de planos apresentada em [17]. Embora o artigo original assuma uma entrada de dados que representa superfícies planares, este trabalho investiga se a divisão do espaço é razoável para encontrar aproximações de superfícies no contexto de estimação de normais.

É importante notar a diferença desta abordagem para aquela do cálculo de planos tangentes em cada vértice, como apresentada pelo Hoppe. Enquanto o algoritmo original aproxima planos na região próxima de cada ponto utilizando uma análise local da vizinhança, a proposta de subdividir o espaço em planos aproximados analisa o conjunto inteiro de pontos da nuvem, dividindo-o em conjuntos menores até que se encontre uma aproximação aceitável por um plano.

Este trabalho utiliza uma octree como estrutura de divisão do espaço. Cada nó da árvore representa uma região do espaço, que pode conter pontos da nuvem. Enquanto a raiz é a região que delimita toda a nuvem de pontos, os nós filhos são sub-regiões de seus respectivos pais.

O critério de subdivisão desta octree é baseado no cálculo de componentes principais [13] (PCA)e da análise da relação entre os autovalores da matriz de covariância associada. Como visto anteriormente, é importante que o elipsóide não esteja deformado a ponto de que seja impossível detectar um plano. Portanto, para detectar se um conjunto de pontos pode ser aproximado por um plano, os critérios ( $\lambda_2 > s_{\alpha}\lambda_3$ ) e ( $s_{\beta}\lambda_2 > \lambda_1$ ) devem ser verdadeiros, onde  $\lambda_1 \ge \lambda_2 \ge \lambda_3$  são os autovalores e os parâmetros  $s_{\alpha}$  e  $s_{\beta}$  são fatores de tolerância. Esses parâmetros foram designados como  $s_{\alpha} = 6,25$  e  $s_{\beta} = 1,5$ , o que quer dizer que o eixo maior não pode ser mais do que 1,5 vezes o eixo médio e este tem que ser 6,25 vezes maior do que o menor eixo. É de se esperar que o menor eixo seja sempre muito menor que os outros dois, cujos tamanhos devem ser parecidos. Abaixo, o pseudo-código da divisão.

#### Algoritmo 1 Algoritmo de subdivisão

**Require:** vertexList

1: *centroide* = Centroide de *vertexList* 

2: cv = Matriz de covariância de vertexList em relação a centroide

3:  $\lambda_1 \ge \lambda_2 \ge \lambda_3 =$  autovalores de cv

4: se  $!((\lambda_2 > s_{\alpha}\lambda_3) \in (s_{\beta}\lambda_2 > \lambda_1))$  então

5: Subdivide recursivamente

6: **fim se** 

A aproximação dos planos tangentes em cada ponto  $x_i$  da nuvem se comporta de forma similar àquela apresentado pelo Hoppe. A diferença é que a vizinhança  $Nbhd(x_i)$ é o conjunto de pontos que pertencem à célula da octree que o vértice  $x_i$  pertence, e o centro  $o_i$  do plano tangente é o próprio vértice  $x_i$ . Pode-se, também, utilizar o cálculo do Hoppe, considerando que apenas os vértices na mesma célula são elegíveis para a vizinhança  $Nbhd(x_i)$ . Abaixo, o pseudo-código do cálculo das normais dentro das folhas da octree.

Algoritmo 2 Cálculo das normais de vértices dentro das folhas da octree					
Require: vertexList					
1: para $x_i$ em $vertexList$ faça					
2: $Nbhd = Vizinhança de x_i$					
3: $centroide = Centroide de Nbhd$					
4: $cv = Matriz de covariância de Nbhd em relação a centroide$					
5: $x_i.normal =$ autovetor associado ao menor autovalor de $cv$					
6: fim para					

#### 5.4.2 Cálculo de consistência dos sentidos das normais

O cálculo de consistência original do algoritmo Hoppe foi utilizado para concluir esse passo do algoritmo. Como a solução aproximada para calcular a orientação dos planos tangentes assume determinadas propriedades da entrada de dados, modificar este passo do algoritmo ainda é um desafio para aumentar a robustez do algoritmo. As alternativas encontradas são para casos particulares que, por sua vez, assumem diferentes propriedades da nuvem de dados. Portanto, por hora, a solução utilizada ainda é a mais geral e robusta encontrada.

Um caso especial interessante é quando o dispositivo de captura retorna a informação da posição de captura  $v_p$ . Neste caso a solução para a orientação de normais é trivial, pois pode-se considerar que o lado da superfície na direção do dispositivo de captura é o lado correto da orientação das normais. Portanto, todas as normais do objeto podem ser orientadas no sentido em direção à posição de captura de modo que satisfaçam  $n_i (v_p - x_i) > 0$ .

### 5.4.3 Estimação de direções híbrida

O uso de uma octree para dividir o espaço gera uma situação indesejada nas bordas das células da mesma. Pelo modelo de subdivisão utilizado, é possível que células adjacentes contenham planos tangentes muito parecidos, especialmente nas regiões das bordas das células.

Como a octree se subdivide em células iguais, este processo não analisa o contexto da amostra dentro de cada célula para efetuar a subdivisão. Isto quer dizer que não é calculado um plano ótimo de corte para dividir o espaço de acordo com as direções dos planos tangentes.

Para mitigar este problema, foi concebido um modelo de estimação de direções híbribo, uma combinação entre o algoritmo de Hoppe e o algoritmo proposto. A ideia é utilizar o calculo do algoritmo proposto apenas para pontos a uma certa distância do centro da célula. O demais pontos são próximos da borda, portanto o cálculo proposto não pode ser utilizado. Para estes, é utilizada a estimação do Hoppe. O resto do algoritmo segue conforme descrito nesta seção.

## Capítulo 6

## **Resultados computacionais**

## 6.1 Introdução

Os algoritmos apresentados foram implementados na linguagem C++. Na comparação da estimação de normais, foi utilizada a implementação da CGAL [1] para testar o algoritmo de Hoppe [12]. A configuração do sistema utilizado para os testes foi um Windows 7, com processador DualCore 2800 MHz e 2GB de RAM DDR3. Aproximadamente, nesta máquina, a estimação das normais do *Stanford Bunny* é obtida em 10 segundos. Já a superfície implícita é obtida em torno de 1 segundo.

As instâncias *Stanford Bunny*, *Happy Buddha* e *Dragon*, utilizadas para os experimentos, foram obtidas através do banco de dados *The Stanford 3D Scanning Repository* [2].

A nuvem de pontos do *Stanford Bunny* consiste em 35947 pontos, com normais recuperadas através de exportação do *Blender*.



Figura 6.1: Imagem original do *Stanford Bunny*, retirada de [2].

A nuvem de pontos do Happy Buddha consiste em 543652 pontos, com normais re-

cuperadas através de exportação do *Blender*. Neste trabalho, foi utilizada uma resolução menor da instância, com 32328 pontos.



Figura 6.2: Imagem original do Happy Buddha, retirada de [2].

A nuvem de pontos do *Dragon* consiste em 437645 pontos, com normais recuperadas através de exportação do *Blender*. Neste trabalho, foi utilizada uma resolução menor da instância, com 22998 pontos.



Figura 6.3: Imagem original do Dragon, retirada de [2].

## 6.2 Descrição dos Experimentos

Os experimentos realizados foram criados para simular modelos que necessitem de uma certa robustez do algoritmo.

### 6.2.1 Simulação de Sombras

Este experimento visou a simulação de sombras na captura do objeto. O modelo é degradado de forma a criar buracos na malha. As degradações seguiram uma medida de percentual de malha degradada. O conjunto de percentuais utilizado foi:  $\{0\%, 1\%, 2\%, 5\%, 10\%, 20\%, 25\%, 30\%, 50\%, 75\%\}$ .

Quatro formas de resultados foram colhidas deste experimento. Todas analisando o erro em graus da normal encontrada em relação à normal correta. O primeiro resultado é uma tabela que analisa o erro mínimo, o erro máximo e o erro médio do conjunto. O segundo é um histograma da freqüência de erros para cada dez graus. O terceiro é uma visualização dos erros das normais no modelo. Por último, é apresentada uma visualização alternativa dos erros das normais, que considera apenas os erros na direção delas, ignorando a orientação.

### 6.3 Resultados

Abaixo, o resultado de uma reconstrução, com partição de unidade e funções de base radial, utilizando as estimações de normais do algoritmo de Hoppe e do algoritmo proposto.

Observando as falhas na orelha do coelho, é possível notar que o algoritmo de Hoppe apresenta alguns erros na estimação de normais, porque a nuvem não respeita os requerimentos do algoritmo na relação entre a densidade e o tamanho da característica do modelo.



Figura 6.4: Reconstrução com estimação de normais de Hoppe.

O algoritmo proposto tem um resultado similar ao algoritmo clássico. Surgem algumas áreas de erro para este tipo de amostra.



Figura 6.5: Reconstrução com estimação de normais do algoritmo proposto.

### 6.3.1 Resultados do Stanford Bunny

### 6.3.1.1 Tabela de mínimos, máximos e médias

Degradação (%)	Hoppe			Octree			Híbrido		
Degradação (70)	min	max	med	min	max	med	min	max	med
0	0,002	83,943	5,523	0,002	116,184	8,163	0,003	138,455	9,674
1	0,002	83,943	5,512	0,005	116,184	8,089	0,002	138,455	9,552
2	0,002	83,943	5,508	0,005	116,184	8,061	0,002	138,455	9,496
5	0,002	83,943	5,503	0,005	$154,\!377$	8,032	0,002	124,063	9,405
10	0,002	169,710	5,529	0,005	$127,\!809$	8,059	0,002	178,741	9,400
20	0,003	179,992	7,767	0,008	179,903	8,601	0,003	179,992	10,908
25	0,003	179,992	9,964	0,008	179,933	10,197	0,004	179,992	12,863
30	0,003	179,992	10,715	0,008	179,809	9,898	0,007	179,992	13,330
50	0,016	179,992	12,267	0,008	179,940	16,008	0,006	179,992	14,213
75	0,070	179,992	71,265	0,008	179,909	53,846	0,070	179,992	72,311

Tabela 6.1: Resultados do Stanford Bunny

## 6.3.1.2 Histogramas



Figura 6.6: Histograma de erros entre a normal estimada e a original para Stanford Bunny com 0% de degradação.



Figura 6.7: Histograma de erros entre a normal estimada e a original para Stanford Bunny com 10% de degradação.



Figura 6.8: Histograma de erros entre a normal estimada e a original para Stanford Bunny com 20% de degradação.



Figura 6.9: Histograma de erros entre a normal estimada e a original para Stanford Bunny com 30% de degradação.



Figura 6.10: Histograma de erros entre a normal estimada e a original para Stanford Bunny com 50% de degradação.

#### 6.3.1.3 Estimações com orientação



Figura 6.11: Comparação entre estimações com 0% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.12: Comparação entre estimações com 10% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.13: Comparação entre estimações com 20% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.14: Comparação entre estimações com 30% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.15: Comparação entre estimações com 50% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.

#### 6.3.1.4 Estimações sem orientação



Figura 6.16: Comparação entre estimações, sem orientação, com 0% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.17: Comparação entre estimações, sem orientação, com 10% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.18: Comparação entre estimações, sem orientação, com 20% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.19: Comparação entre estimações, sem orientação, com 30% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.20: Comparação entre estimações, sem orientação, com 50% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.

## 6.3.2 Resultados do Happy Buddha

Degradação (%)		Hoppe		Octree			Híbrido		
	min	max	med	min	max	med	min	max	med
0	0,002	179,838	26,634	0,002	179,825	29,261	0,003	179,997	59,821
1	0,007	179,838	26,890	0,005	179,825	29,568	0,003	179,997	59,927
2	0,007	179,838	27,152	0,006	179,995	$102,\!138$	0,003	179,997	59,979
5	0,008	179,935	28,054	0,013	179,994	106,764	0,000	179,992	52,315
10	0,008	179,838	29,402	0,020	179,996	109,021	0,002	179,997	64,161
20	0,008	179,690	27,466	0,010	179,996	96,836	0,002	179,995	55,223
25	0,044	179,690	24,226	0,010	179,996	107,431	0,002	179,994	66,270
30	0,069	179,730	33,091	0,004	179,975	42,761	0,004	179,995	79,890
50	0,073	179,931	74,000	0,006	179,992	78,011	0,004	179,995	90,525
75	0,137	179,585	82,975	0,049	179,969	79,779	0,002	179,986	86,019

### 6.3.2.1 Tabela de mínimos, máximos e médias

Tabela 6.2: Resultados Happy Buddha

### 6.3.2.2 Histogramas



Figura 6.21: Histograma de erros entre a normal estimada e a original para Happy Buddha com 0% de degradação.



Figura 6.22: Histograma de erros entre a normal estimada e a original para Happy Buddha com 10% de degradação.



Figura 6.23: Histograma de erros entre a normal estimada e a original para Happy Buddha com 20% de degradação.



Figura 6.24: Histograma de erros entre a normal estimada e a original para Happy Buddha com 30% de degradação.



Figura 6.25: Histograma de erros entre a normal estimada e a original para Happy Buddha com 50% de degradação.

#### 6.3.2.3 Estimações com orientação



Figura 6.26: Comparação entre estimações com 0% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.27: Comparação entre estimações com 10% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.28: Comparação entre estimações com 20% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.29: Comparação entre estimações com 30% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.30: Comparação entre estimações com 50% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.

#### 6.3.2.4 Estimações sem orientação



Figura 6.31: Comparação entre estimações, sem orientação, com 0% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.

180



Figura 6.32: Comparação entre estimações, sem orientação, com 10% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.33: Comparação entre estimações, sem orientação, com 20% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.34: Comparação entre estimações, sem orientação, com 30% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.35: Comparação entre estimações, sem orientação, com 50% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.

## 6.3.3 Resultados do Dragon

Degradação (%)	Hoppe			Octree			Híbrido		
	min	max	med	min	max	med	min	max	med
0	0,026	179,541	13,777	0,013	$179,\!896$	27,044	0,001	179,998	26,937
1	0,026	179,541	13,766	0,009	$179,\!973$	$31,\!596$	0,000	179,892	24,001
2	0,026	179,541	13,836	0,009	$179,\!973$	$31,\!680$	0,002	179,892	$24,\!259$
5	0,061	179,541	14,060	0,005	$179,\!945$	$23,\!332$	0,002	179,997	24,799
10	0,076	179,451	14,368	0,003	$179,\!997$	$23,\!921$	0,002	179,992	$25,\!677$
20	0,076	179,451	15,193	0,003	$179,\!979$	$25,\!285$	0,002	179,995	31,577
25	0,102	179,451	15,600	0,002	$179,\!994$	34,842	0,002	179,995	31,226
30	0,098	179,528	16,465	0,005	$179,\!997$	28,713	0,002	179,992	30,681
50	0,100	179,886	24,779	0,004	179,994	52,142	0,002	179,997	58,885
75	0,102	179,858	91,265	0,005	179,984	83,489	0,003	179,995	91,754

### 6.3.3.1 Tabela de mínimos, máximos e médias

Tabela 6.3: Resultados Dragon

#### 6.3.3.2 Histogramas



Figura 6.36: Histograma de erros entre a normal estimada e a original para Dragon com 0% de degradação.



Figura 6.37: Histograma de erros entre a normal estimada e a original para Dragon com 10% de degradação.



Figura 6.38: Histograma de erros entre a normal estimada e a original para Dragon com 20% de degradação.



Figura 6.39: Histograma de erros entre a normal estimada e a original para Dragon com 30% de degradação.



Figura 6.40: Histograma de erros entre a normal estimada e a original para Dragon com 50% de degradação.

#### 6.3.3.3 Estimações com orientação



Figura 6.41: Comparação entre estimações com 0% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.42: Comparação entre estimações com 10% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.43: Comparação entre estimações com 20% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.44: Comparação entre estimações com 30% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.45: Comparação entre estimações com 50% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.

#### 6.3.3.4 Estimações sem orientação



Figura 6.46: Comparação entre estimações, sem orientação, com 0% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.

180



Figura 6.47: Comparação entre estimações, sem orientação, com 10% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.48: Comparação entre estimações, sem orientação, com 20% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.49: Comparação entre estimações, sem orientação, com 30% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.



Figura 6.50: Comparação entre estimações, sem orientação, com 50% de degradação. Na esquerda o algoritmo do Hoppe, no centro o algoritmo Octree proposto e na direita o algoritmo Híbrido.

### 6.4 Análise dos Resultados

O erro médio do algoritmo de Hoppe é menor do que dos algoritmos propostos. Pelos histogramas é possível verificar que a freqüência de normais cuja o erro é menor do que 20 graus é mantida pelo algoritmo do Hoppe, enquanto o algoritmo de octree encontra um número grande de normais com erro próximo de 180 graus. O algoritmo híbrido tem bons resultados, mas não o suficiente para melhorar os resultados de Hoppe.

A estrutura de octree do algoritmo proposto gera uma vizinhança descontínua nas bordas de cada célula. Isto é interessante apenas em regiões onde foram identificados planos tangentes diferentes. No entanto, nas células adjacentes cujos planos tangentes dos pontos próximos à borda têm angulação muito próxima, esta separação insere erros no cálculo das normais. A vizinhança destes pontos na borda não inclui pontos que pertencem à outra célula, embora estes pontos sejam interessantes para compor a vizinhança.

Os casos com muitas normais que apresentam erro próximo a 180 graus são ocasionados pelo segundo passo do algoritmo, aquele que calcula a orientação das normais. Este cálculo se baseia na diferença em graus de um plano tangente e seus adjacentes para montar uma estrutura de propagação. Portanto esta propagação é influenciada pelas diferenças inseridas pela octree.

E importante notar que os erros mais graves são aqueles que se aproximam de 90 graus, pois indicam que a estimação de direção das normais não foi capaz de encontrar um plano tangente aceitável para determinado ponto.

## Capítulo 7

## Conclusão e trabalhos futuros

Este trabalho estudou uma melhoria para a estimação de normais apresentada no trabalho de Hoppe [12]. O algoritmo proposto dividiu o espaço de acordo com a planaridade das regiões da nuvem de pontos, afim de aumentar a robustez em relação ao algoritmo de Hoppe. A idéia de dividir o espaço é promissora. Ao dividir o problema de reconstrução de superfícies em problemas menores, pode ser possível encontrar soluções melhores com precisão maior em diferentes resoluções. No entanto, a utilização de uma *octree* para esta estratégia se mostrou inadequada. O principal problema desta estrutura é o seu método de subdivisão.

Como a *octree* se subdivide em oito partes iguais, certas células podem possuir planaridade parecida com suas células vizinhas. Nas bordas entre este tipo de célula, a estimação é incorreta, pois a vizinhança utilizada no cálculo da orientação das normais deixa de contemplar pontos que fazem parte da vizinhança geodésica. Idealmente, para substituir a *octree* deveria ser utilizada uma estrutura de divisão do espaço que fosse capaz de se subdividir de acordo com o contexto da planaridade da superfície a ser reconstruída. É possível notar que a árvore geradora mínima criada no segundo passo do algoritmo é sensível às mudanças no primeiro passo: uma pequena diferença em vetores normais pode causar uma propagação incorreta, como nos resultado do *Buddha*.

Duas alternativas foram concebidas para trabalhos futuros. A primeira alternativa utiliza a *octree* e, depois da subdivisão, mescla células vizinhas cuja planaridade geral seja muito parecida. Isto transformaria conjuntos de células com planaridade parecida e próximas no espaço em uma célula apenas, evitando os problemas nas bordas. A segunda alternativa seria utilizar uma estrutura com uma subdivisão consciente das diferenças de planaridade na amostra, algo baseado em uma árvore *BSP*. Quanto à detecção de planos, seria interessante investigar a utilização de um cálculo alternativo ao *PCA*. Um ajuste de superfícies polinomiais pode ser uma escolha melhor para nuvens de pontos que representem objetos com poucos planos [8]. Tal ideia é comumente usada para estimação de propriedades diferenciais em superfícies aproximadas por malhas (superfícies lineares por partes). A etapa inicial do método aqui proposto pode ser compreendida como um aprendizado linear de manifolds simplificado. Em trabalhos futuros, pretendemos investigar como outras formas de aprendizado de manifolds [20] podem auxiliar no problema de estimativa de normais, por exemplo, usando regularização laplaciana.

## Referências

- The Computational Geometry Algorithms Library (CGAL), última visita em 2016. Disponível em http://www.cgal.org/.
- [2] The Stanford 3D Scanning Repository, última visita em 2016. Disponível em https: //graphics.stanford.edu/data/3Dscanrep/.
- [3] ALLIEZ, P.; JAMIN, C.; MÉRIGOT, Q.; MEYRON, J.; SABORET, L.; SALMAN, N.; WU, S. CGAL 4.7 - Point Set Processing, 2015. Disponível em http://doc.cgal. org/latest/Point\_set\_processing\_3/.
- [4] ALLIEZ, P.; SABORET, L.; GUENNEBAUD, G. CGAL 4.7 Surface Reconstruction from Point Sets, 2015. Disponível em http://doc.cgal.org/latest/Surface\_ reconstruction\_points\_3/.
- [5] BERGER, M.; TAGLIASACCHI, A.; SEVERSKY, L. M.; ALLIEZ, P.; LEVINE, J. A.; SHARF, A.; SILVA, C. T. State of the art in surface reconstruction from point clouds. *EUROGRAPHICS star reports 1*, 1 (2014), 161–185.
- [6] BOULCH, A.; MARLET, R. Fast and robust normal estimation for point clouds with sharp features. In *Computer graphics forum* (2012), vol. 31, Wiley Online Library, pp. 1765–1774.
- [7] 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 (2001), ACM, pp. 67–76.
- [8] CAZALS, F.; POUGET, M. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design 22*, 2 (2005), 121–146.
- [9] DEY, T. K.; LI, G.; SUN, J. Normal estimation for point clouds: A comparison study for a voronoi based method. In *Point-based graphics*, 2005. Eurographics/IEEE VGTC symposium proceedings (2005), IEEE, pp. 39-46.
- [10] EDELSBRUNNER, H.; MÜCKE, E. P. Three-dimensional alpha shapes. ACM Transactions on Graphics (TOG) 13, 1 (1994), 43–72.
- [11] HOPPE, H. Surface Reconstruction from Unorganized Points. Tese de Doutorado, University of Washington, Seattle, WA 98195, 1994.
- [12] HOPPE, H.; DEROSE, T.; DUCHAMP, T.; MCDONALD, J.; STUETZLE, W. Surface reconstruction from unorganized points. In *Computer Graphics (Proc. SIGGRAPH)* (1992).

- [13] JOLLIFFE, I. Principal component analysis. Wiley Online Library, 2002.
- [14] JU, T.; LOSASSO, F.; SCHAEFER, S.; WARREN, J. Dual contouring of hermite data. ACM Transactions on Graphics (TOG) 21, 3 (2002).
- [15] KAZHDAN, M.; BOLITHO, M.; HOPPE, H. Poisson surface reconstruction. In Proc. of the EG/SIGGRAPH Symposium on Geometry processing (2006).
- [16] LI, B.; SCHNABEL, R.; KLEIN, R.; CHENG, Z.; DANG, G.; JIN, S. Robust normal estimation for point clouds with sharp features. *Computers & Graphics* 34, 2 (2010), 94–106.
- [17] LIMBERGER, F. A.; OLIVEIRA, M. M. Real-time detection of planar regions in unorganized point clouds. *Pattern Recognition* 48, 6 (2015), 2043–2053.
- [18] LIN, C. H.; CHEN, J. Y.; SU, P. L.; CHEN, C. H. Eigen-feature analysis of weighted covariance matrices for lidar point cloud classification. *ISPRS Journal of Photogrammetry and Remote Sensing* 94 (2014), 70–79.
- [19] LORENSEN, W. E.; CLINE, H. E. Marching cubes: A high resolution 3D surface construction algorithm. ACM siggraph computer graphics 21, 4 (1987), 163–169.
- [20] MA, Y.; FU, Y. Manifold Learning Theory and Applications, 1st ed. CRC Press, Inc., Boca Raton, FL, USA, 2011.
- [21] OHTAKE, Y.; BELYAEV, A.; ALEXA, M.; TURK, G. Multi-level partition of unity implicits. ACM Trans. Graph. (Proc. SIGGRAPH) (2003).
- [22] TOBOR, I.; REUTER, P.; SCHLICK, C. Efficient reconstruction of large scattered geometric datasets using the partition of unity and radial basis functions. *Journal of* WSCG (2004), 467–474.
- [23] TURK, G.; O'BRIEN, J. F. Variational implicit surfaces, 1999. Technical Report GITGVU-99-15.