ACADÉMIE DE MONTPELLIER

Université Montpellier II

SCIENCES ET TECHNIQUES DU LANGUEDOC

# Thèse

En vue de l'obtention du grade de

DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER II

Discipline : **Génie Informatique**, **Automatique et Traitement du Signal**
Formation Doctorale : **Systèmes Automatiques et Microélectroniques**
Ecole Doctorale : **Information**, **Structures**, **Systèmes**

Présentée et soutenue publiquement par

**Carla Silva Rocha AGUIAR**

Le 24 Septembre 2009

## SEGMENTATION AND REGION-BASED REGISTRATION APPLIED TO 3D RAW POINT CLOUDS. APPLICATION TO CULTURAL HERITAGE

**Jury** (Provisoire)

| | | | |
|---|---|---|---|
| Rapporteurs | El Mustapha | MOUADDIB | *Professeur, Université de Picardie Jules Verne* |
| | Sylvie | PHILIPP-FOLIGUET | *Professeur, ENSEA* |
| Examinateurs | Samir | AKKOUCHE | *Professeur, Université de Lyon 1* |
| | Peter | STURM | *Directeur de Recherches, INRIA Grenoble* |
| Directeur de thèese | André | CROSNIER | *Professeur, Université Montpellier II* |
| Co-directeur de thèse | Sébastien | DRUON | *Maître de Conférences, Université Montpellier II* |

# Abstract

As 3D geometry has gained increasing popularity as the new form of digital media content, it has seen an increasing number of projects to acquire detailed 3D representations of cultural heritage objects at museums and archaeological excavations. The goals of acquiring 3D digital models of cultural heritage objects are to improve preservation, archiving, understanding, restoration, and dissemination these artworks. To accomplish these goals, new scanning devices have been developed to capture fine details, irrespective of the size and number of objects, and supplementary information, such as color and texture. The output of these acquisition systems in most of the time is a raw point cloud, a noisy approximation to the continuous object surface, consisting of a set of points coordinates.

High-precision scanning devices produce raw point cloud composed of large datasets, up to millions of points [LPC+00] and they acquire many raw point clouds, each from a single viewpoint. This data does not become usable until the relative viewpoints of each point cloud is brought to the same reference frame and then merged into a final 3D digital model. The process of identifying and matching corresponding regions across multiple point clouds, given arbitrary initial positions, and estimating the corresponding rigid transformation that best align the point clouds to each other, is called *registration*.

This thesis aims at understading and evaluating how large raw point clouds can be efficiently represented and processed in the context of registration. Much of the work presented in this thesis is part of the EROS-3D project, which aims at dealing with the management of large 3D artwork object collections. We will examine the data representation problems for the registration of *scans* representing cultural heritage objects at museums. As the volume of data increases, the current registration techniques are not well adapted to treat this problem, and the overload of pre-processing becomes enormous. A compact representation of this data becomes an unavoidable and challenging task. This strategy is appealing because, by focusing attention away from the entire input data to small regions, we place ourselves in a better position to handle practical challenges arising from raw point clouds such as missing data, or occlusion, noise, outliers, and other objects in the scene. We are particularly interested in the development of a

registration technique that works directly with the unstructured raw data and with its compact representation, without any other pre-processing.

While developing a complete 3D registration system with raw point cloud as the modelling primitive is beyond the scope of this document, this dissertation presents progress towards some fundamental building blocks that are crucial for achieving this goal.

- To provide a proper context for our work, we give a brief introduction to the process of modelling from reality in Chapter 1. We also describe the main cultural heritage projects to introduce the particular datasets with which this thesis is concerned, and to expose some of the particular challenges in developing tools to process these datasets. Finally, we explicit the problems we will address in this thesis.

- In Chapter 2 we present efficient methods for characterising the point clouds. We present the fundamental properties of the underlying continuous surfaces, and how these properties lead to assumptions about the raw point cloud in terms of noise level, points distribution, and about the shape around a small neighborhood. To our knowledge, there are no local shape descriptors estimation algorithm designed to work directly on raw point clouds. We extend two curvature estimation methods [FJ89, Tau95], originally designed to work on triangle mesh surface representation. Both methods estimate the curvature at a point through the analysis of the curves deviation in very close neighborhood of this point. We derive the equivalent between these two approaches and we evaluate the accuracy of such descriptors on both synthetic and real noisy datasets.

- In Chapter 3, we investigate the possibility of using region as the data representation. Region reduces the amount of data considerably, while keeping the complete input information. A raw point cloud can be partition into regions through a process called segmentation. We propose a graph-based segmentation algorithm [ADC07a, ADC07b] that partitions large, dense, and noisy point clouds into connected regions respecting some homogeneity predicate. We also propose a mathematical formulation of the regions obtained from this technique. To validate our algorithm, extensive experiments are done to evaluate the segmentation and it motivated the use of region, as data representation, to perform global registration.

- In Chapter 4 we investigate the problem of pairwise global registration. We introduce a global registration algorithm [ADC09] based on region correspondence, an efficient scheme for raw point clouds, which is resilient to noise and outliers. This approach aims at solving the problem of comparing inconsistent, noisy features, or comparing only features that are not in the overlapping area. The algorithm is evaluated using real raw point clouds.

- Finally, in Chapter 5 we conclude with a discussion of some key points developed in this thesis and some extensions, improvements to the algorithms proposed to serve as a potential starting point for future work in this area.

# Acknowledgments

This thesis is the result of my work at LIRMM whereby I have been accompanied and supported by many people. Here is my opportunity to express my gratitude for all of them.

First I would like to express my sincere thanks to my parents, Valter and Nailza Aguiar, for always being by my side, for giving me a good education, and encouraging me to pursue my dreams. From them I have gotten a love of knowledge and the desire to balance both professional and personal life. I owe it to them for any good values that I have in myself.

I would like to thank my advisors Sébastien Druon and André Crosnier for their support and inspiration during my stay at LIRMM. Thank you for your patience through the many ups and downs over the years. I feel lucky to have been able to learn from them and work with them on a variety of exciting topics. I also want to thank all people I worked with during the PhD. Marie-Jose Aldon, who encouraged and motivated me during my first year of PhD. All members of EROS 3D project, who shared their insights, their opinions about my work, and were generous to exchange their data and algorithms.

I wish to also thank Geovany Araujo Borges for his invaluable guidance during my undergraduate studies, for helping to shape the way I conduct my research and for being an outstanding example. Jean Triboulet, Philippe Poignet and Etienne Dombre for giving me the first opportunity to study abroad. Also my dearest friend Flavio Vidal who has constantly given me professional, personal and musical advices through my career.

I express my many thanks to all my friends at the LIRMM for the good and bad times we spend together leaving me with lots of fond memories to cherish. Kevin Loquin (for the uncountable theaters and coffees together), Michel Dominici (for being the greatest listener and for your always great advices), Abdellah El Jalaoui, Jean-Mathias Spiewak (for your good humor and jokes), Andreea-Elena Acunta (for trying hard to teach me how to dance), Ashvin Sobhee (for your insights into the broader picture), Vincent Bonnet(for organizing the "South of France" parties), Nicolas Riehl (for your good advices in movies and lifestyle), Mourad Benoussaad (for our priceless conversations at lunch), Peter Meuel (for sharing with me about the trials and tribulations of being a graduate student), Olivier Parodi (for keeping me company at the lab until late at night and always scaring me with your unexpected appearances), Philipe amat (for sharing your codes and ideas), Milan Djilas (for your humor and encouragement into an

academic career), Mitsu, Arturo Gil Pinto, Guta, Gabriel Marchesan, Floor Campfens, Walid Zarrad, Baptiste Magnier, Sébastien Cotton, Maria Papaiordanidou, Sébastien Durand, Lama Mourad, Khizar Hayat, Zafar Shahid, Hai Yang.

I wish to also thank my sisters and my family, each one in their own way, for supporting me all this time. In particular, my grandmothers, for making a great effort to understand my work, my aunt Margarete Rocha, for the phone therapies and encouragements, my aunt Iracema Regina, for her constant emails and visit, my aunt Marcia Rocha, for our funny Sunday phone chats, my cousins Luiza Lauro and Jonas I, for their excitement and enthusiasm towards my work. My sisters, Viviane and Juliana Aguiar, who were always there for me, supportive and incredibly understanding despite my long absences. Thank you two for being the my best friends, my confidents, and my travelling mates.

Finally, I want to thank to all friends that were my family these years. They made my life happy, exciting and fun, and without them I would have had many more stressful and worrisome moments. I express my sincere gratitude to Beatrice Reynaud, who is an incredible generous person that welcomed me into her family, and always knew the right words to say. A special thanks to Rogerio Richa (Roger), who is an excellent source of all kinds of information. In particular his knowledge of computer vision were invaluable. Besides, he taught me the value of a friendship and the importance of sharing and counting on others. Thanks to him, I improved my research, social, and personal skills and I can say that he became a brother to me. I feel also truly blessed to have met so many outstanding friends that influenced my work and kept me relatively sane. My gratitude and love to Antonio Bo (the best roommate ever, my lifestyle model, my conscience, my safe port), Daniella Pingret (my dearest xuxuzinha, my listener, my wing man, my biggest fan and always in the front row, my organization), Akane Saiki (my lovely Penguin, my intuition, my peace), Thais Salum (also the best roommate ever, my cooking teacher, my shopping mate, my vocabulary teacher), Thaina Corvacho (my beautiful, carrying, Tchuca, my daily phone calls, I hope we can do everything we planned together), Bruno Vilhena (my idol, my master, my greatest adviser), Polyana Vilhena (my dance mate, my cooking goodness, my travel company), Alice Vilhena (my xodozinha), Rosimeri Silviero (my stunt mother, my wonderful breakfasts, my motherhood model), Pedro Paim (my proud, my example of professionalism, my one-in-a-lifetime friend), Hiromi(my guide, and I hope one day I will be like you), Thiago Resende(my "friend since 7th grade", my spiritual guide, my guitar hero), Gisela Cabral (my Giselinha, always beside me no matter the distance between us), Vinicius Araujo, Alexandre Martins, William Miranda, Marcio Alexandre, Antonio Geraldo, Ilana Marques, Flavia Queiroz, Francisco Silviero, Claudia Lanvagne, Marcus Chafim, Talita Rosa, Leticia Taira, Alexandra Szajner, Ludovic Molle, Damien Bauduffe, Pierre Garcia, Rodrigo Teixeira, and to all other friends.

# Contents

# List of Figures

# Chapter 1

# Modeling From Reality

## 1.1 Introduction

The advance of 3D scanners in the last decades has consolidated the 3D geometry as a digital medium. In the present, it is possible to accurately capture three-dimensional (3D) geometry of real objects, modify, and analyze the shape of their 3D digital models efficiently. As a result, growing repositories of 3D data are available, encouraging the development of new tools to process the geometry, the texture and the complexity of such data. In Figure 1.1 we illustrate some 3D digital models, where the richness of levels of details, supplementary information, and nature of the objects vary from one model to another.

Many application domains benefit from these 3D digital models. In archaeology, 3D digital models of artworks and historical monuments are used to permanent archive, perform diagnostic test and verify the state of conservation, create digital libraries and produce faithful physical replicas of such models. Other domains that can profit from modelling from reality are special effects [ZSCS04], virtual reality [GSC+07, HBK+09, GCG06, GDH+00], reverse engineering and rapid prototyping [HM01], civil engineering [PMW+08] and robot navigation [LMA07].

Applications exploit different aspects of 3D digital models. One example in archaeology is showed in Figure 1.2, where the geometric entities are 3D digital models of one statue and one mould set. The problem consists in finding the mould in which the statue was made from. The use of 3D digital models in this application is particularly relevant, since these artworks can be fragile, of difficult manipulation, and they can be located physically in different museums. To find a matching measure between one mould and the statue, the comparison of the geometry of the surfaces on both models are needed. Comparing the global aspects (global shape matching) of one mould and one statue does not give a good similarity measure, since most of the moulds and statues have the same topology. High similarity between particular

**Figure 1.1:** *Examples of 3D digital models. Different applications and domains have distinct requirements about the aspects of the 3D models. These examples illustrate different levels of details, sampling accuracy, supplementary information such as color and texture.*

features in both models, on the other hand, enhances the match, since wrong matches are less frequent. This feature approach agrees with an intuitive notion that a strong matching between particular areas on both mould and statues is much more significative than a uniquely global matching. Thus, the matching between a statue and a mould should take into account only feature matching of particular features in both mould and statue. The best matches obtained using the 3D digitalized models can be then check by an archaeologist, in order to find the correct correspondence. Another application showed in Figure 1.3, concerns a 3D search engine for indexing and retrieving 3D digital models from artwork database [GCJ+07]. In this application, it is performed the indexing between the same statue set, but it takes into account global and local aspects of the 3D model.

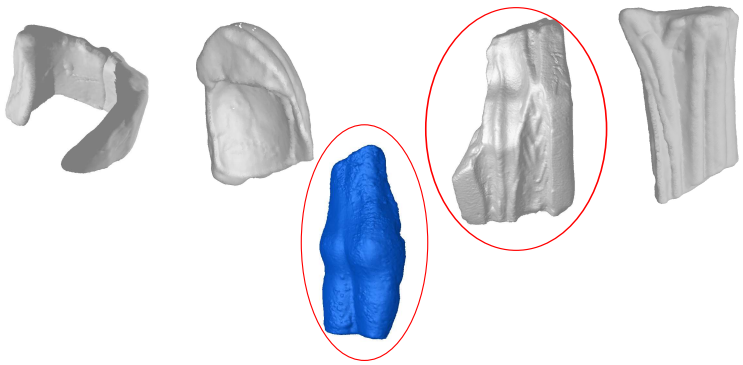


**Figure 1.2:** *The use of 3D digital models to find statue and mold correspondence [ED].*

The context of this thesis is the *modelling from reality*, which consists in generating a 3D digital model from 3D measurements performed on a physical *scene* using a 3D scanning device. In the next section, we will present some of the efforts done to make the *modelling from reality*

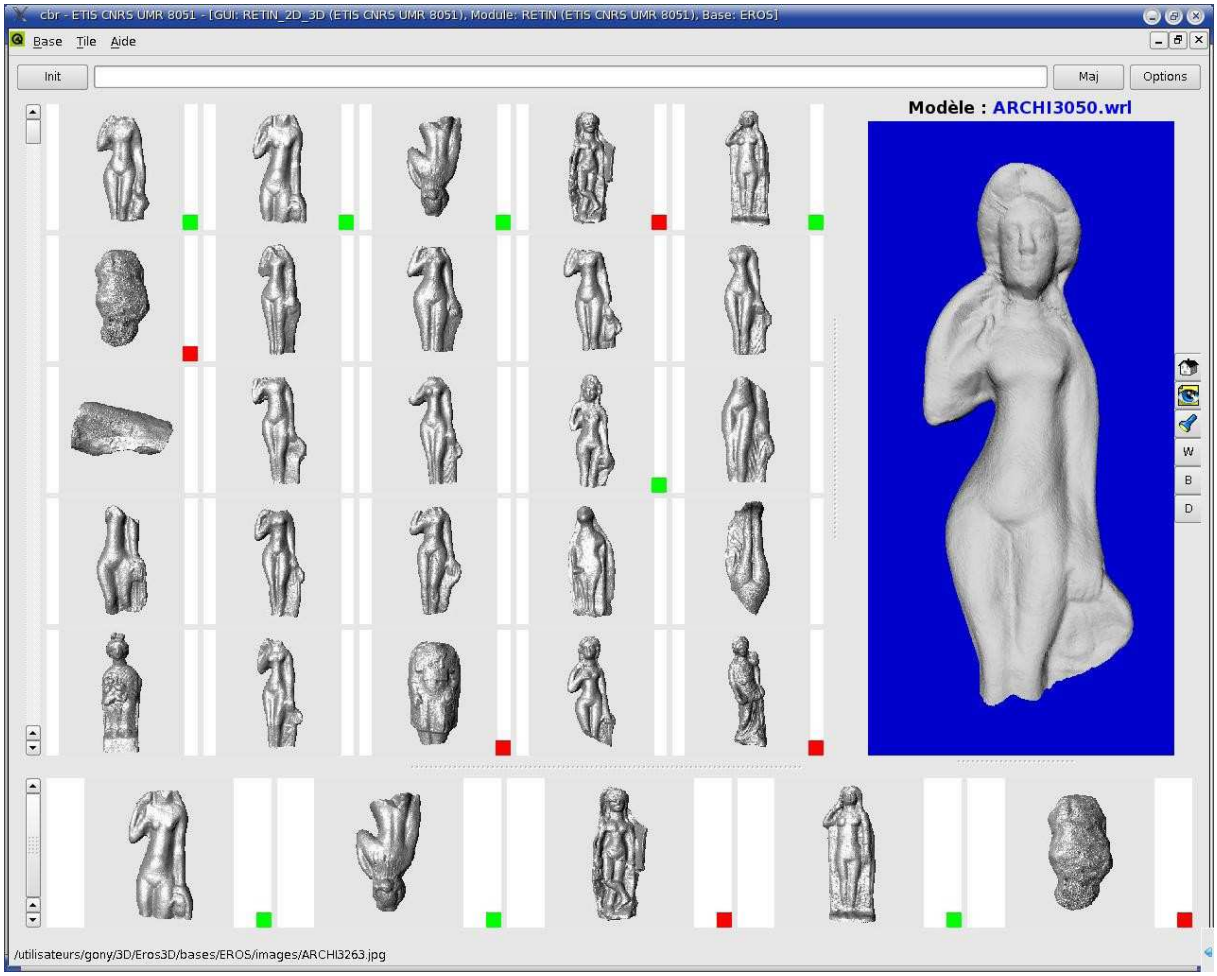



**Figure 1.3:** *3D-content-based retrieval in an artwork database [GCJ+07].*

more accessible to a larger range of applications.

## 1.2 Review of some projects relating to modelling from reality

There has been a large effort to develop new tools, techniques and systems to allow an efficient acquisition and manipulation of 3D media. Advances in 3D scanning technologies permit the acquisition of large variety of *scenes*, with both big and small objects present in it, with a modest amount of manual labor. High-precision scanners permit to acquire small features on the *scene*, with a very little noise. The availability of new tools allow an efficient visualization [RL00], editing [ZPKG02, lib, 3DR], retrieving [FKMS05] and indexing [APFJ+08a] of 3D digital models. An large number of 3D media repositories are available [Repa, Sch, Repb, pr, Dat], and most of these repositories are accessed just by browsing through images captured from the stored 3D geometry.

The bottlenecks in both scanning and manipulating 3D media techniques depend mainly on the application, and the improvements are usually focused in a specific application or domain. Much of the work presented in this thesis is part of the EROS-3D project, which aims at dealing with the management of large 3D artwork object collections. We will examine the data

representation problems for the registration of *scans* representing cultural heritage objects at museums.

An increasing number of projects have been seen to acquire detailed 3D representations of cultural heritage objects at museums and archaeological excavations, with a goal of improving preservation, understanding, restoration, archiving, and dissemination. Cultural heritage projects present a variety of practical constraints: statues cannot be moved, and must be scanned *in situ*, so the scanner had to be transportable. For safety reasons, the scanner must stay a certain distance from the statues at all times. Since the error in triangulation scanners is proportional to the viewing volume, the combination of high resolution and large object size imposes very tight calibration constraints. As the resolution increases, the number of *scans* per objects increases, and registration algorithms adaptive to this large data volume is required. Visualization and editing tools that handle this amount of data are also necessary. In this context, we will list some of the main projects, the problems that propose to solve, the strategy used and the results achieved.

### 1.2.1 Artwork acquisition projects

(a)

(b)

**Figure 1.4:** *Artwork acquisition projets. (a) The Digital Michelangelo [LPC+00] and Pietà [BRM+02] projects.*

The Digital Michelangelo project [LPC+00] is one of the earliest large-scale 3D scanning projects in the cultural heritage domain. It has as technical goal to make a 3D archive of Michelangelo's statues. It also had the particular challenge of capturing Michelangelo's chisel marks and the surface color. In order to capture detailed chisel marks, both for analysis and

to accurately render self shading effects, it was necessary to acquire geometry at roughly 0.25 *mm* resolution. To accomplish this goal, they developed a hardware and a software system to digitalize and visualize the shape and the color of these statues. They designed a laser triangulation scanner, based on Cyberware's commercial systems, to capture the chisel marks, and acquired color in a separate pass across the object. They also developed a software to align the *scans* to the same reference frame. Their acquisition pipeline provided digital models with high resolution, and they were able to capture the chisel marks, and the alignment software was showed to be relatively robust. However, this pipeline is time consuming, and requires some operator intervention. In this project, advances were made in scanning large objects under non-laboratory conditions, in local [RL01] and multi-view [Pul99] registration of large datasets, in the capture of color, and in the visualization of large datasets [RL00]. Problems that were not solved include the global registration, that was performed by an operator, and some issues concerning the acquisition system calibration [Bro08]. A similar acquisition project was later proposed to digitalize the Pietà and some other Michelangelo statures [BRM+02].

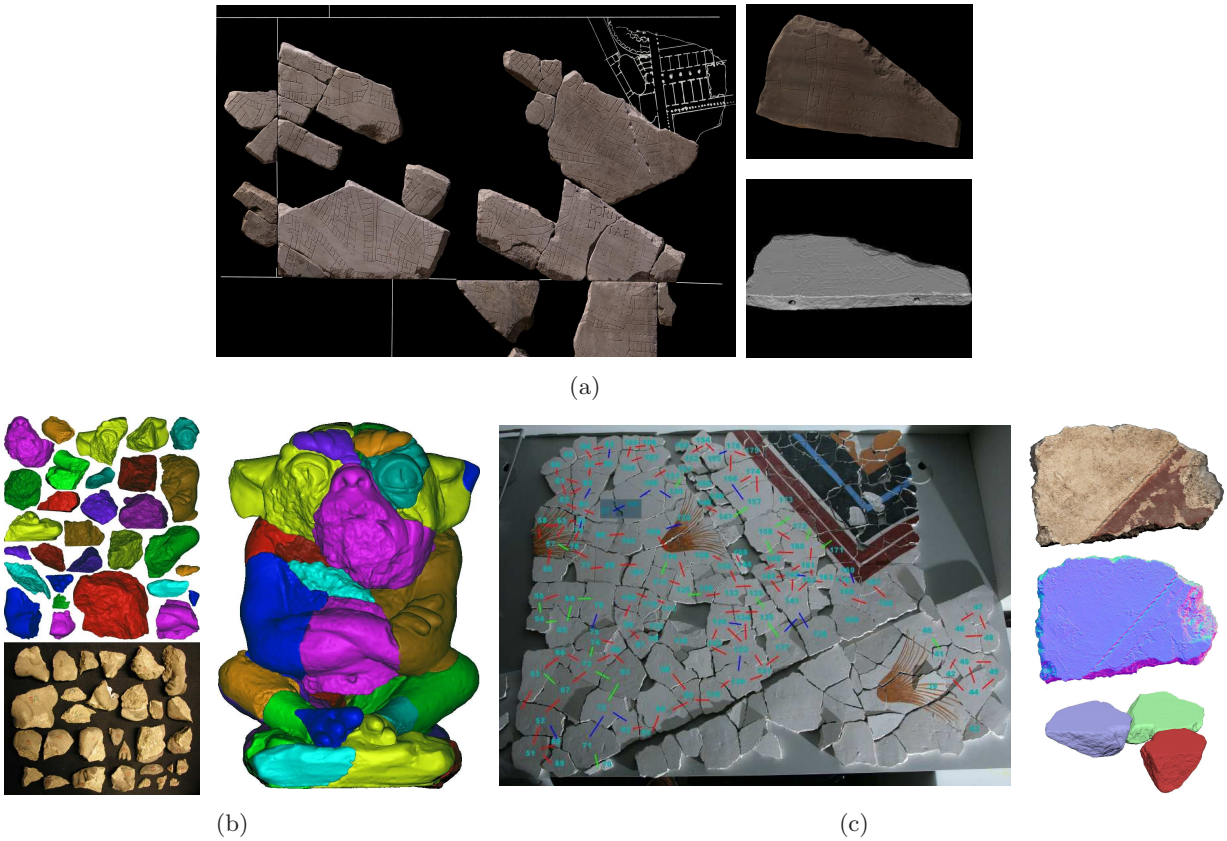


**Figure 1.5:** *Assembling fragments projects. (a) Forma Urbis Romae project [Pro], (b) 3D puzzles project [HFG+06], and (c) Theran Fresco project [Bro08].*

In conjunction with the Digital Michelangelo project, the same team undertook to scan all extant fragments of the *Forma Urbis Romae* [Pro, KL05], a marble map of Rome from the early 3rd century A.D. The map is a blueprint of Rome, including not only building, but rooms, doorways, and even staircases, all of them marked by incisions less than 1 *mm* deep. Approximately 10% of the map's surface is presently known, and is broken into 1273 fragments of varying size

and thickness. Efforts have been underway to assemble these fragments, and thereby learn as much as possible about ancient Rome. For example, it was developed a technique in which is possible to extend the incisions associated with roads and buildings to find matches between non-adjoining fragments [KL05]. This supplementary goal was later formulated as a matching problem [HFG+06]. Similar to the *Forma Urbis Romae*, there is the Theran Fresco [Bro08], and the 3D puzzles [pr, HFG+06] projects. The Theran Fresco project is a collaboration between the Akotiri Excavation, Thera and Princeton University, and it concerns the technical aspects of the acquisition, alignment and matching systems that are designed to aid in reconstruction of frescoes at the site. In this projects, they concentrated their efforts to solve the problem of the alignment [BR07] and assembly [BTFN+08] problems in cultural heritage. They also developed an inexpensive and rapid 3D model acquisition system [BTFN+08] that produces complete models of at least 10 fragments per hour with a single non-expert operator. In Figure 1.4 we show examples of the digital 3D models obtained in each project.

### 1.2.2 Automatic Reconstruction Conduit project

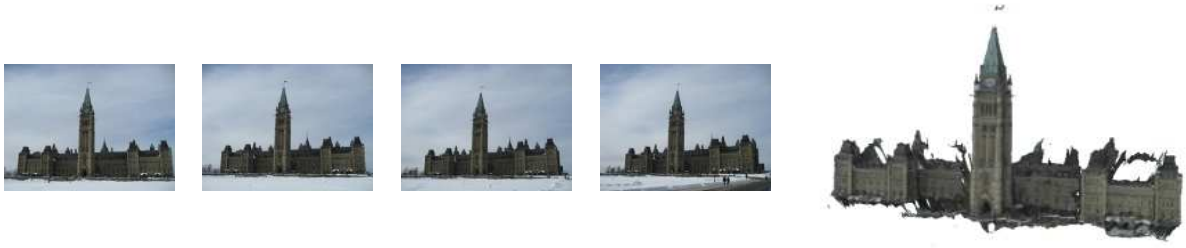


**Figure 1.6:** *The Automatic Reconstruction Conduit (ARC) project [VG06]. From an images set of the same environment, it is obtained the corresponding 3D digital model.*

The Automatic Reconstruction Conduit (ARC) project [VG06, Con] has a very different philosophy, compared to other projects concerning the acquisition of 3D digital models of cultural heritage objects. Instead of using scanning acquisition devices, the ARC project ambitions to obtain a 3D model from a sequence of photos of an object, taken from different viewpoints, different cameras and different lightning conditions. Two partners of the *European Network of Excellence* (EPOCH), the *ESAT-PSI* lab of K.U.Leuven (Belgium) and the Visual Computing Lab of *CNR-ISTI* (Italy) decided to set up a low-cost 3D reconstruction pipeline to be used in the cultural heritage field. The idea is to provide an online tool that upload users photos from a given object and output the corresponding 3D model, obtained from this images set, as showed in Figure 1.6.

The technical challenges in these projects are mainly in the domain of 2D image processing. Global image comparison and matching, pairwise and projective triplet matching and the self-calibration, Euclidean reconstruction and dense matching, are among the problems that were treated in order to robustly obtain a correct 3D model. The advantage of this system is that users can exploit home made photos to obtain 3D digital models, which helps with the popularization of 3D media. However, the accuracy and the resolution of the 3D digital object is enough only

for visualization, and their use to research purpose is not appropriated.

### 1.2.3 EROS-3D project

**Figure 1.7:** *EROS-3D project. Some 3D digital models from the EROS-3D database, from statues fragments, to vases and complete statues.*

The EROS-3D project [GCJ+07] aims at dealing with the management of large 3D artwork object collections. The C2RMF (Center for Research and Restoration of the Museums of France)[fRotMoF] organizes for several years campaigns of digitalization in museums from all France. The EROS-3D is a collaboration between the C2RMF and the following french laboratories: *Laboratoire d'Informatique, de Robotique et de Microlectronique de Montpellier* (LIRMM), *Laboratoire d'Informatique en Image et Systèmes d'information* (LIRIS), *Laboratoire Electronique, Informatique et Image* (LE2I), and *Equipe de Traitement des Images et du Signal* (ETIS). The objective of the project is to develop a software architecture to obtain the digital model from *scans*, store, display, retrieve, and compare these data with various levels of use.

This project aims to acquiring artworks of medium size at many different museums, each of which can be completely scanned on commodity hardware in minutes. The supervision consists of placing the artwork on a turntable, clicking a button, turning the object over when it has been completely scanned so that both the front and back surfaces are acquired, clicking another button, and removing the object from the turntable. To this end, it was used the commercial Minolta acquisition system and the first technical challenge addressed in this project consists in the registration of the *scans* obtained from the acquisition. We contribute with the project in this stage of the modelling from reality process, were a set of *scans* must be brought to a common reference frame.

At this time, the EROS-3D database contains 650 objects digitalized as very high definition 3D models. Each model contains from 100.000 to 3.000.000 points. These objects mainly are figurines of the Gallo-Roman civilization ($-100$, $-300$ after JC): Mother Divinity, Venus, moulds, vases and many fragments. The C2RMF pays a particular attention to two categories of objects of the base, figurines representing Mother Divinities and those representing Venus. An example of the 3D digital models used in this project is showed in Figure 1.7. The technical challenges

that was considered in this project concerned the registration of the *scans* [DAa06, ADC09], the reconstruction [MCAG07], the streaming [ACA07], the indexing [APFJ+08b] and the retrieving [GCJ+07] of these digital models. At a first moment, the tools were firstly dedicated to historians and archaeologists, to help them with the task of manipulating, comparing and visualizing the artworks from the database. Though, this database is not intended to be reserved only for professionals. In the future, museum visitors provided with their PDA for instance will have the opportunity to ask the EROS-3D database in front of a statue and thus obtain additional information.

## 1.3 Modelling from reality pipeline

The process of *modelling from reality* is illustrated in Figure 1.8. The term *scene* refers to both objects and environments. In the most general setting, 3D scanning devices measure the 3D structure of a *scene* from a single viewpoint. They capture detailed 3D geometry of visible area of the *scene* and they provide much more accurate depth information than conventional vision cues, such as motion, shading, and binocular stereo [HTZ04]. The output of this acquisition procedure is a raw point cloud, a noisy approximation to the continuous *scene* surface, consisting of a set of points coordinates.

Generally, one *scan* is not enough to model the entire *scene*. Occlusions, limitations in the sensor's field of view and material reflective properties cause missing data and holes in the sampling, since these areas are unobserved by the sensor. Data are missing for objects at infinite distance like the sky, and metal, glass and ceramic surfaces where the laser rays never return to the scanner [HTZ04]. Though, when seen the same *scene* from different points of view, the unobserved areas may become apparent. Therefore, to form a complete 3D digital model of a *scene*, one must sample the *scene* from multiple viewpoints and then combine these *scans*. The process of identifying and matching corresponding regions across multiple *scans*, given arbitrary initial positions, and estimating the corresponding rigid transforms that best align the point clouds to each other, is called *registration*.

Once all *scans* are combined and brought to the same reference frame, there is the integration phase, where the registered *scans* are combined into a unified representation. Additional processes, such as denoising, filtering, hole fitting, surface reconstruction and meshing, modify the point cloud in order to remove noise, outliers and redundant samples in the overlapping areas.

The goal of this thesis is to understand and evaluate how the dataset can be efficiently represented in the context of *scans* registration. As the volume of data increases, the current registration techniques are not adapted to treat this problem, and the overload of pre-processing becomes enormous. We focus on the development of features to represent the dataset and reduce the data volume. This way, the current registration techniques can be used to the alignment of large datasetsets.

A fundamental question in the design of such algorithms is the choice of the underlying modelling primitive. Most of *modelling from reality* systems use triangular meshes as their

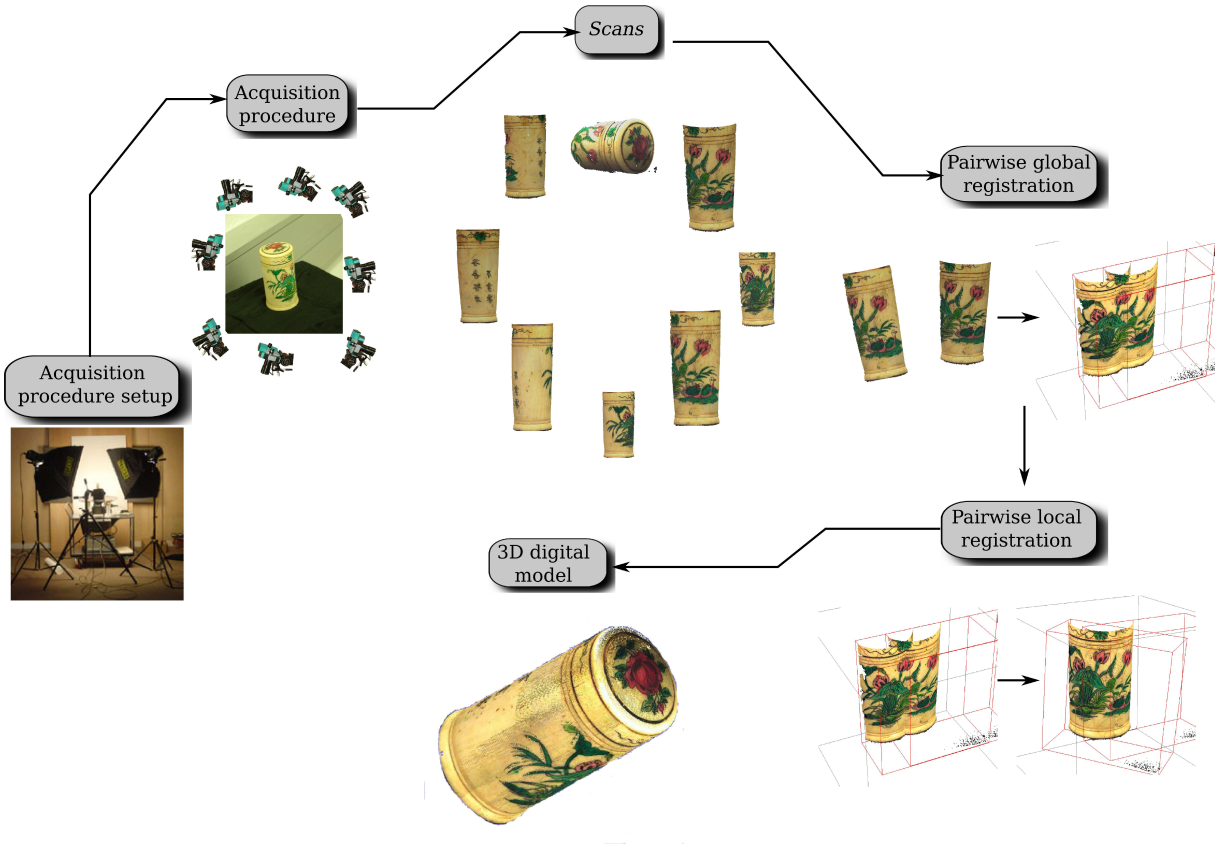



**Figure 1.8:** *Modelling from reality process. From a continuous real object to its digitalized model.*

modelling primitive. Meshes are undirected graphs composed of a collection of vertices and edges, whose faces represent discrete piecewise approximations of surfaces. They are preferred because they explicit points connectivity and normals. However, meshes are normally not the output of a 3D sensor. The construction of a mesh from noisy data is not straightforward and requires denoising and additional pre-processing that can undesirably remove high frequency features. Some alternative 3D model representations exist, like range images, point clouds, and parametric surfaces (e.g., NURBS), but they are not usually used by registration algorithms. We are particularly interested in the investigation of registration techniques that work directly on the unstructured raw data, without any pre-processing step. We consider that data pre-processing, like reconstruction, denoising, and filtering, must be performed after all point clouds are aligned in the same reference frame and the complete 3D model is obtained, and not before registration.

Before we specify more about data representation of large, raw point clouds, we will examine the registration stage of the modelling from reality pipeline.

## 1.4 Pairwise *scans* registration

*Registration* is the process of identifying the overlapping area across different *scans*, in arbitrary initial positions, and estimating the corresponding rigid transformations that best align the *scans* to each other. It is assumed that exists, for each *scan*, at least another *scan* where the

surface from one *scan* is a subset of the other. If such assumption is not verified, erroneous alignments occur.

The registration can be seen as the problem of searching for correspondences, followed by a transformation estimation and alignment evaluation. Given two *scans* of the same object, the goal is to find correspondences between the *scans* and compute the transformation that brings the two *scans* into alignment.

A registration algorithm is generally classified broadly into three classes: global, local and multi-view registration. What differentiates one problem from another are the strategy adopted to find correspondences and estimate the alignment, the amount of data considered in these computations, and the error metric adopted to verify and validate the results.

Global, or coarse, registration problem consists in finding a rough initial *scans* alignment without any prior knowledge of the relative spatial positions of the *scans*, or the overlapping area. This initial *scans* positioning was for a long time performed manually. As the number of *scans* per object grows, it becomes an arduous task, and the availability of global registration algorithms that automatically estimate the initial *scans* alignment aided to the popularization of 3D geometry.

The global registration problem is particularly difficult because there is no information about the initial position of the *scans*. Besides, the input *scans* contain noise and outliers, and the shapes overlap only over parts of their extent. Most of the proposed methods [HFG+06, GMGP05, HCH00, MGGP06, Mit06, JH99] benefit from the low dimension of rigid transformations, and use local descriptors to extract a small set of points correspondences, sufficient to estimate the transformation.

Once the *scan* pair is approximately aligned, the transformation refinement is applied in order to minimize the distance between overlapping surfaces. Local, or fine, registration algorithms [Zha94, BM92, Bro92, AMCO08, CM92, RL01, PHYH06] perform such refinement. The best-known methods for local registration are variation of the Iterative Closest Point (ICP) algorithm [Zha94, BM92]. ICP is an iterative procedure minimizing the mean square error between corresponding points across the point clouds. It consists in matching points in both point clouds and then estimates the rigid transformation that minimizes the distance between corresponding points. This procedure is repeated iteratively until its convergence or until it reaches a certain number of iterations. However, as the volume of data increases, this task becomes both time and memory consuming, and, thus, unfeasible to large datasets.

When more than two *scans* are involved and the initial pose estimates are given, the process is called multi-view registration. Multi-view registration algorithms [Pul99, HH01, HP05] are often used to distribute pair-wise registration errors evenly over an entire model. The multi-view registration problem will not be dealt with in the present thesis.

An automatic registration algorithm commonly performs in sequence global, local and multi-view registration in order to generate the complete 3D digital model. Despite the numerous proposed methods to solve the problem of pairwise registration in Computer Graphics and Computer Vision, there are still several problems that prevent a surface registration from *scans* in a robust, accurate and efficient way, given a *scan* pair in arbitrary position. The main

limitations of most pairwise registration methods are the following:

- *Global registration robustness.* When currently available global registration algorithms are applied to partially overlapping surfaces in the presence of noise and missing data, they usually tend to be unstable and inefficient [HP05].

- *Local registration convergence.* Local registration algorithms suffer from slow convergence and shallow local minima in the presence of "difficult" geometry, like flat, spherical and symmetric surfaces. These surfaces contain relatively few features constraining the alignment [GIRL03] and they can cause local minimum convergence or divergence of ICP algorithm and its variants, even for good initial alignment. This is caused by the use of too many points coming from featureless regions in the alignment transformation estimation. An effort has been done by ICP variants to treat some/all of these error sources, but they are application specific [GLB01], or make some assumptions about the surface and the data structure [JH99, GIRL03].

- *Mesh approaches.* We stress out that most of currently available local and global registration algorithms work on triangle mesh surfaces. To obtain the mesh representation, the raw data are heavily pre-processed to filter and structure the dataset. The capability of such methods to work directly on raw point clouds is not straightforward [AMCO08]. Meshing and filtering processes increase the complexity of the registration pipeline, and smooth high frequency features. They should be done once the registration process is finished.

- *Feature-based approaches.* Both local and global pairwise registration problems rely strongly on finding correct correspondence between two *scans*. In most cases, *scans* overlap only in parts, and the correspondence can be seen as a *partial matching* problem. In such scenario, matching *scans* using of global descriptors fails, because global properties might change considerably between them. The solution is to rely on the geometry of the common part [Mit06], and it becomes challenging as the common parts are unknown. The majority of registration methods use local shape descriptors at a point to represent the input *scans*, and to find point correspondences. However, local shape descriptors are sensitive to noise, and, therefore, they are not a robust *scan* representation.

- *Not applicable to large datasets.* Heavy pre-processing steps and the use of a large portion of the input dataset make most of the registration algorithms unfeasible to large datasets. To solve this dimensionality problem, some authors propose to re-sampled the input dataset to reduce the data volume [Pul99, GIRL03], or use discriminant features to represent efficiently the dataset [HFG$^+$06, GMGP05, JH99]. All these pre-processing do not guarantee that the reduced data will have features in the overlapping area.

The most important and most critical stage of registration algorithms is find correct correspondence between `scans` without knowing the topology or the geometry of the surface represented by the point cloud. To find the set of corresponding points in two *scans*, the existing

*scans* registration algorithms focus their efforts in analyzing the input shapes separately and, after some processing. To achieve robustness, most algorithms perform an excessive data pre-processing. However, not only this pre-processing does not guarantee the robustness, since it is performed indepently to both *scans*, but it also limits the applicability of such algorithm to large data volume. The motivation of this thesis is to provide data representation that allows the registration process to work directly on the raw point cloud furnished by the acquisition system. We propose a region representation to solve these problems. The input data, instead of being represented by a feature set, is represented by a region set. It provides a compact, complete and geometrically relevant information about the *scene* surface. It naturally reduces the volume of data without loosing information about the dataset. It also minimizes the noise effect, once it does not rely on individual point information. Therefore, region descriptive power is higher relative to other features. Finally, since region space is much smaller than feature space, we can exploit efficiently the entire search space to solve the global registration problem. Before we present our region-based registration scheme, we will discuss the advantages and drawbacks of using the raw point cloud as modelling primitive over triangle mesh in the *modelling from reality* pipeline.

## 1.5 Raw point clouds as modelling primitive

There has been a resurgence in the use of point clouds [KB04] as the modelling primitive for 3D processing. Levoy and Whitted [LW85] initially proposed point clouds as geometric rendering primitives, and since then it has been used in many domains and applications [KV03, UH07, HDD+92, AK04, MAVdF05, KB04, UH08, ZPKG02, DMH08, DMH08]. However, very few registration approaches use raw point cloud as their modelling primitive.

One popular philosophy adopted by registration methods is to pre-smooth, re-sample the point clouds and obtain a mesh representation twice, first before registration is performed and later when after all `scans` are aligned. The smoothing and meshing as a pre-processing to registration results in more stable feature estimation. Consequently, the performance registration algorithms based on feature correspondence is improved. Finally, once all point clouds are in the same reference frame, data integration is neccessary to obtain the complete 3D digital model and remove redundant information in the overlapping area between *scans*, and thus, the same smoothing, re-sampling and meshing are applied.

The influence of pre-processing on the raw point cloud before and after registration is illustrated in Figure 1.9. In Figure 1.9 is showed two complete 3D digital models after registration and intergration is performed. The differences between models representing the same object is due to the fact that registration was performed using two different strategies. In Figures 1.9 (a),(c), the *scans* were previously smoothed, and after that, the registration algorithm in [GMGP05] computed the alignment transformation. The registered *scans* are again smoothed in the integration phase and the final 3D model is showed in Figure 1.9 (a),(c). In Figure 1.9 (b),(d), the alignment transformation was computed directly on the raw point cloud, using the algorithm presented in [AMCO08], and then smoothing applied to the registered models in the

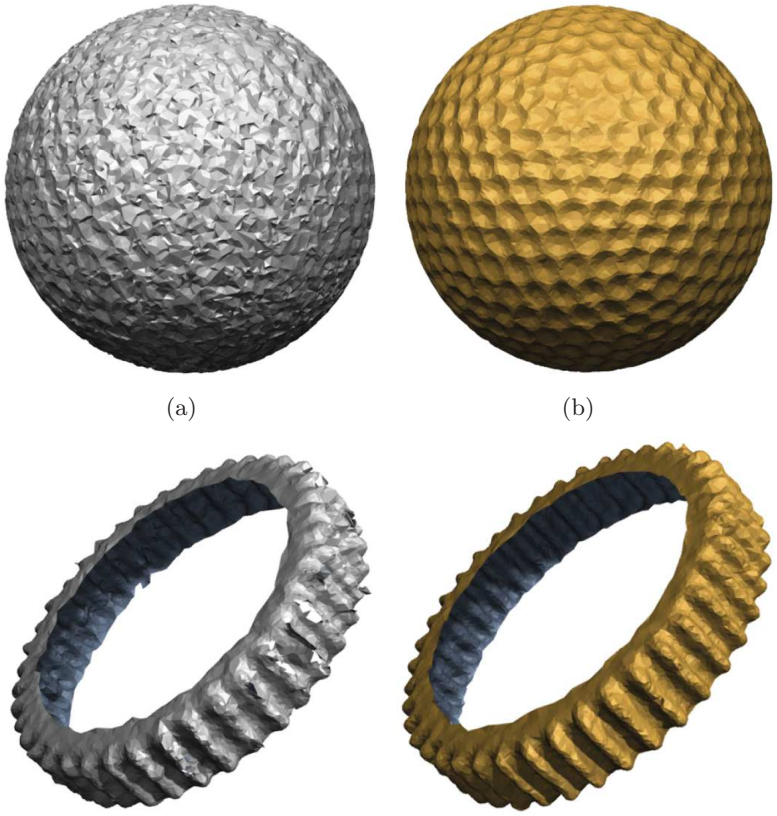



**Figure 1.9:** *Registration of raw point clouds [AMCO08]. Complete 3D digital models, after performing registration of the scans and integration of the complete model. (a) and (c): complete models when smoothing the point clouds before and after registration. (b) and (d): Models when smoothing the point clouds only after registration.*

integration phase. In both cases, the same operator is used for surface smoothing. Comparing the models obtained from these two approaches, we observe that, when the smoothing is a pre-processing to registration, high frequency features are degenerated and the final model does not recover the original object geometry 1.9 (a),(c).

This step of denoising the raw point cloud results in a removal of noisy samples, but it also results in a significant loss of high frequency features [AMCO08]. It compromises the performance of most registration algorithms, which use local shape descriptors. Besides, because of the increasing complexity of 3D models, there is an associated overhead of processing and managing connectivity information in meshes. In such scenario, it is considerably more efficient to work directly on raw point clouds rather than polygons [Pau03]. Through this thesis, all the presented techniques are implemented with points as a unified acquisition, processing and rendering primitive.

In order to incorporate the point cloud as our modelling primitive, we make a few assumptions about the point cloud and the underlying surface from which it was sampled. To be practically useful, these assumptions must be sufficiently general to be widely applicable, but sufficiently concrete to allow the algorithm to use them effectively. Through this thesis, we will make the following assumptions concerning the point points:

- The point cloud, a geometric model of a *scene*, is a noisy approximation of a piecewise smooth surface. This surface is a "connected, possibly with boundary, embedded in $\mathbb{R}^3$. By piecewise smooth we mean that such surface is 2-manifold (homeomorphic to $\mathbb{R}^2$), except on sharp features, such as *creases*, *corners* and *darts*. These assumptions hold true for most of real *scenes*, from man-made objects to natural environments.

- Usually, for 3D scanner devices, the acquisition is performed in a regular sampling lattice attached to the sensor reference frame, not on the surface reference frame. Consequently, point clouds lack of any regular lattice-like structure, and, globally, the point cloud is *irregularly* sampled, in the sense that points do not follow a known regular sampling distribution. Even though such assumption holds true, we assume that, in a small neighborhood, the point clouds are *regularly* distributed.

- The size of the smallest feature, or level of detail, captured by the point cloud is determined by sample density, as defined by Dey in [Dey06].

- Noise model is unknown. Point clouds acquired using scanning sensors are corrupted with a non-anisotropic noise. The noise is highly directional and depends on the distance of the point to the sensor [UH07]. In addition, noise model depends on the sensor used to acquire the point and the acquisition procedure. Otherwise confidence associated with the measures can be given, we do not consider any noise model or make any hypothesis about points distribution (usually non-Gaussian [BMG94]).

The raw point cloud is then a discrete, noisy approximation of a continuous surface, where neither the topology nor the geometry of the surface represented by the point cloud are known in advance. A compact representation of this dataset becomes an unavoidable and challenging task. In the next section, we place the above challenges in the context of registration using region as data representation. We then list our contributions towards meeting these challenges along with an outline of the document.

## 1.6 Motivating problem

The driving motivation behind our work is to provide data representation that allows performing registration directly on large raw point clouds. As discussed before, feature estimation in presence of significant noise and outliers still remains a challenging task. In such scenario, we propose a region-based scheme, where the input point cloud is represented by a region set and the registration is performed in region space, instead of a feature space. A simple illustration of this concept is showed in Figure 1.10. In Figure 1.10 (a), we have two raw point clouds that overlap partially. In Figure 1.10 (b) both point clouds are represented by a set of compact surfaces. These regions are subsets of the raw point clouds. Each region denotes a portion of the *scene*, with size and shape depending on the scale and on the surface model. By visual inspection, the correspondence between regions in the overlapping area is well established, even for partially overlapping regions (Figure 1.10 (c)). Observe that the data considered in this stage

is considerably reduced, compared to the input dataset. Finally, the regions in correspondence can then be used to compute the initial alignment transformation that brings both *scans* to the same reference frame (Figure 1.10 (d)).

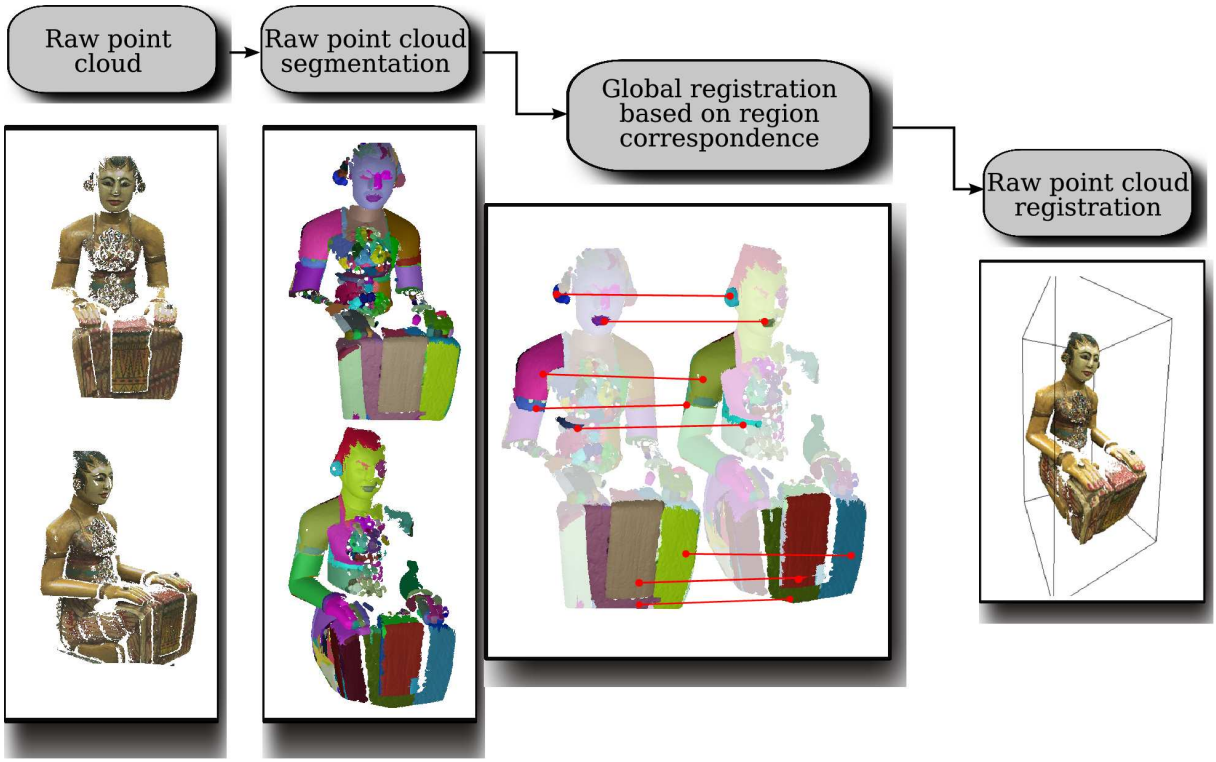



**Figure 1.10:** *Region-based registration pipeline.*

This strategy is appealing because, by focusing attention away from the entire input data to small regions, we place ourselves in a better position to handle practical challenges arising from raw point clouds such as missing data (occlusion) and other objects in the *scene* (clutter).

The use of region as data representation introduces different challenges to the registration pipeline. First, the construction of reliable regions from the raw point cloud is of great practical value. The regions must verify some known homogeneity predicate, so the region extraction process produces predictable results. Since geometric aspect is what makes 3D models an appealing environment representation, region homogeneity predicate should rely on geometric properties. To this matter, local shape descriptors must be estimated from the raw point cloud to characterize the local shape around each point. Shape descriptors can be used to form and describe regions. Even if such estimation is not robust to noise, re-sampling and missing data, the region extraction process must adequately capture this variation in local shape descriptors, and points connectivity.

Once regions are extracted, registration can be seen as the problem of finding region correspondences between *scans* that overlap. To compare efficiently regions in different *scans*, regions descriptors should be as invariant as possible to changes in sensing conditions and invariant under rigid transformation. Even if the area that overlaps and information about the shape of the *scans* are unknown *a priori*, a registration algorithm must identify and remove wrong cor-

respondences. This is a particular challenge of global registration problem, when the *scans* are in arbitrary initial positions.

In what follows, we will formulate the problems we want to solve, the strategy we propose to solve them and how we incorporate region representation into a region-based registration algorithm.

### 1.6.1 Raw point cloud segmentation

The core of this thesis is to represent a point cloud as a set of regions. The problem of partitioning the data into meaningful regions is called segmentation. It is usually an intermediate phase, in which objective is mostly a substantial reduction in data volume and to use segmented regions in higher-level processing. Several applications using 3D digital models and 3D raw point clouds, like reverse engineering [HM01], indexing [APFJ+08a], retrieving [GCO06], recognition [BAAC08], simplification [Pau03], region-based editing [CGR+04], and registration [HP05], can benefit from regions obtained from segmentation.

Most *scenes*, like man-made objects (buildings, desks), animate objects (human and animals), and free-form objects (trees and terrain), cannot be modelled only by elementary parametric surfaces. The main difficulty in developing and evaluating segmentation algorithms is then to establish a definition for meaningful region. Different definitions yield to distinct but at the same time correct results [UPH07, HJBJ+96].

In this thesis, we are interested in a data driven segmentation algorithm, since we want to partition free-form surfaces. Specifically, we want to partition an unstructured point cloud into significative regions, in multiple scales and different sizes. It aims at reducing the data volume, structuring the dataset and extracting higher-level surface information from the point cloud. At the core of our approach is the belief that data-driven point grouping can be used to extract equivalent regions in the overlapping area of different *scans*.

We define our segmentation algorithm as a recursive bi-partition of the weighted *neighborhood graph* [ADC07a, ADC07b]. Region homogeneity is inferred from the local feature analysis on a weighted graph constructed from the triplet: Euclidean neighborhood relationship, feature space defined by local shape descriptors and similarity function. We extended the 2D segmentation algorithm using the graph's Minimum Spanning Tree ($MST$), introduced in [Zah71b] and later improved in [FH04], to work directly on raw point clouds.

We formalized the segmentation as a graph bipartition where the region homogeneity is defined in some feature space. However, the raw data is an unstructured point clouds with no local shape information attached to the points. To our knowledge, there are no local shape descriptors estimation algorithm designed to work directly on raw point clouds. We extend two curvature estimation methods [FJ89, Tau95], originally designed to work on triangle mesh surface representation. Both methods estimate the curvature at a point through the analysis of the curves deviation in very close neighborhood of this point and we evaluate the accuracy of such descriptors on both synthetic and real noisy datasets.

At the end of segmentation, we obtain a set of regions. Region properties can be then exploited in a higher level application. We will use region descriptors and adjacency relationships

to develop a region-based global registration algorithm.

### 1.6.2 Global registration based on region correspondence

Given a pair of *scans* in arbitrary and unknown positions, global registration can be divided in two sub-problems. The portions of the two scan that overlap are established through point or region correspondence. Next, using a set of point correspondences in the overlapping area, the estimation of an initial transformation is performed. The result has to be sufficiently close to the correct registration pose in order to be used as the starting pose for automatic fine registration.

Earlier work has identified several promising strategies that could be employed for pairwise scans alignment under these conditions. Nearly all global registration algorithms use a feature-based strategy to find points correspondence. To deal with the growing data volume, salient features on both *scans* are identified and only a smaller sub-sample are used to match features under some rigidity (distance) constraint. The main drawback of feature-based approach is that its performance is directly related to the robustness of feature estimation and salient feature extraction.

From a different perspective, we formulate the global registration algorithm as the problem of finding a set of region correspondences and use these correspondences to compute the initial alignment transformation. In the case of rigid motion, three corresponding points are sufficient to uniquely determine the alignment transform, composing of six parameters. Working in the region space, the alignment transform is determined by three corresponding regions. Assuming that, in the overlapping area, there are at least three corresponding region pairs, a naive alignment scheme has a time complexity of $O(m^3n^3)$: for each region triplet from one *scan*, take a set of three regions from the second *scan*, solve the unique rigid transformation using this correspondence, and evaluate the quality of the current transform. This solution requires computing the transformation for the entire set [GMGP05] and it becomes overly expensive as the number of regions grows.

We propose a novel algorithm to explore efficiently the search space of possible sets of corresponding region triplets. Our region-based global registration algorithm compares the intrinsic properties of regions to establish an initial potential correspondence set, according to regions similarity. We build a decision tree, where each node represents a possible pair of corresponding region triplets. The decision tree represents the solution space of potential correspondences, and it is assigned to each node a matching measure, composed by the region probability distribution function dissimilarity and internal pairwise distance error. A combinatorial search method then explores efficiently the solution space, and find the set of three corresponding regions that minimizes the registration error between two *scans*.

# Chapter 2

# The point cloud

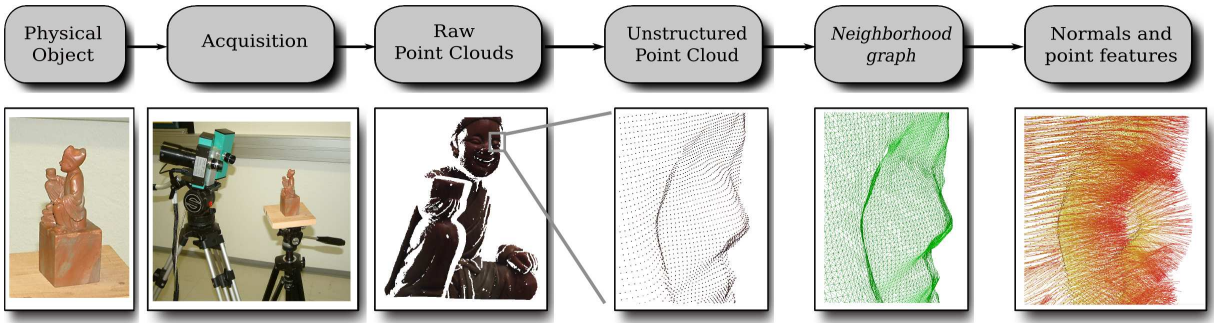



**Figure 2.1:** *From continuous to discrete surface representation.*

## 2.1 Introduction

In this chapter, it is covered the background material for raw point cloud processing in the context of 3D registration and 3D segmentation. We present the fundamental properties of the underlying continuous surfaces, and how these properties lead to assumptions about the raw point cloud in terms of noise level, points distribution, and about the shape around a small neighborhood. We present algorithms to compute two intrinsic properties of the surface. These algorithms process directly the raw point cloud and they estimate the tangent plane and the curvature at a point in a small neighborhood.

Given some physical object, a general 3D acquisition pipeline is illustrated in Figure 2.1. First, a 3D scanning device captures detailed 3D geometry of visible areas of the object [CL96, LPC+00, BTFN+08]. Additional information, such as texture, intensity and other complementary measurements can be also measured. Then, the data coming from different sensors, embedding different types of information, are merged and integrated into a single dataset. This

final dataset is delivered in form of a raw point cloud and it can describe complex geometric object. In general, information regarding the surface is not provided, such as its topology, its geometry, or local shape information, such as normal vectors and curvature at each point. Occlusion, shadow or limitations in the sensor's field of view are translated in the point cloud representation by missing data, or holes. Noise introduced by measurement errors scatters the sample points away from the surface [Dey06] and, alone, point coordinates do not provide neither shape or object topology information, nor surface orientation.

In this Chapter, we present efficient methods for characterising the point cloud. We will represent the point cloud as a *neighborhood graph*. We estimate local surface properties, such as normal vector and curvature, directly from raw point cloud. From the 2-manifold surface in 3-space hypothesis, tangent plane is computed using a plane fitting approach [HDD+92] on a small neighborhood (section 2.5). The normals are then oriented consistently and we propose an algorithm to orient globally the normals of point clouds that are non connected. The curvature at point is estimated using two different methods [FJ89, Tau95], originally designed to work on triangle mesh surface representation (section 2.6). We finally evaluate these algorithms on synthetic and raw point clouds.

## 2.2 From continuous to discrete surface representation

In this section it is presented some of the properties of the continuous surfaces and how these properties are "translated" in their discrete representation. We also describe the necessary assumptions about sampling, noise, feature size to correctly represent the surface.

In a noise-free case, the point cloud is a set of points from a continuous surface $\Sigma$. The surface $\Sigma$ denotes a *scene* composed by a set of sub surfaces that are connected, possibly with boundary, embedded in $\mathbb{R}^3$. A surface $\Sigma$ is said to be connected if any two of its points can be joined by a continuous curve in $\Sigma$. The surface $\Sigma$ is piecewise smooth, which means that it is 2-manifold except on sharp features.

The assumption that the topological space of $\Sigma$ is a 2-manifold is crucial for the algorithms presented in this chapter to compute geometric features. The definition of a 2-manifold (or regular) surface, taken from [dC76], is:

**Definition 2.1** *A subset $\Sigma \subset \mathbb{R}^3$ is a 2-manifold, or regular, surface if, for each $\mathbf{x} \in \Sigma$, there exists a neighborhood $\mathcal{N}$ in $\mathbb{R}^3$ and a map $\pi : U \to \mathcal{N} \cap \Sigma$ of an open set $U \subset \mathbb{R}^2$ such that:*

1. *$\pi$ is differentiable. This means that if we write $\pi(u, v) = (x(u, v), y(u, v), z(u, v))$, for $(u, v) \in U$, the functions $x(u, v), y(u, v), z(u, v)$ have continuous partial derivatives of all possible orders in $U$.*

2. *$\pi$ is a homeomorphism. This means that $\pi$ has an inverse $\pi^{-1} : \mathcal{N} \cap \Sigma \to U$ which is continuous.*

3. *For each $\mathbf{u} \in U$, the differential $d\pi_{\mathbf{u}} : \mathbb{R}^2 \to \mathbb{R}^3$ is one-to-one.*

where the mapping $\pi$ is called a *parametrization* or a *system of coordinates* in a neighborhood of $\mathbf{x}$.

In definition 2.1, condition 1 is necessary to do differential geometry on $\Sigma$. The map $\pi$ is $C^i$-continuous if the $i$th order ($i > 0$) partial derivatives of $\pi$ are continuous. If the partial derivatives of $\pi$ of all orders are continuous, we say $\Sigma$ is $C^\infty$-smooth. In this work, we assume that $\Sigma$ is a piecewise smooth surface. It means that the surface is $C^2$-smooth everywhere, except on sharp features and on the boundary.

Condition 2 has the purpose of preventing self intersection in 2-manifold surfaces. Condition 3 guarantees the existence of a tangent plane at all points of $\Sigma$ and, together with the first, actually enforce smoothness [Dey06]. The surface definition is based on a local neighborhood $\mathcal{N}$, for $\mathcal{N} \to 0$, and all these three conditions imply that the functions like $\pi$ defined in the neighborhood of each point of $\Sigma$ overlap smoothly. The intrinsic properties considered in this chapter and the chapters to follow, such as tangent plane, *directional curvature*, are defined with respect to this neighborhood.

### 2.2.1 Point cloud definition

The point cloud $X$ is defined as the finite set of samples $\mathbf{x}_i \in \mathbb{R}^3$ that approximates to some underlying piecewise smooth surface $\Sigma$:

$$X = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_n\} \subset \mathbb{R}^3 \tag{2.2}$$

As any sample from measurements, the points are assumed to be noisy observations of an unknown underlying surface. For scanned point clouds, Unnikrishnan in [UH07] outlined that noise is almost always highly directional and the noise level of the observation varies as a function of distance of the observed point to the sensor. The noise introduced by the measurement errors scatters the sample points away from the true surface:

$$\mathbf{x}_i = \mathbf{x}_i^0 + \mathbf{e}_i \quad \text{where } |\mathbf{e}_i| < \delta \quad \forall i \tag{2.3}$$

with $\mathbf{x}_i^0 \in \Sigma$ being the unknown true point lying on the unknown surface $\Sigma$ and $\mathbf{e}_i \in \mathbb{R}^3$ being point-dependent error vector, caused by measurement errors.

Other than measurement error, there are also the outliers, caused by measurements at reflective surfaces, or at surfaces near parallel to the sensor $z$ direction. Outliers are spurious points far from the true surface or can be point from a different surface. Usually, they are isolated from the rest of samples and the error magnitude is not bounded as in (2.3).

To capture the error in most sampling processes, we assume that the noise magnitude is bounded and isotropic. It is said that the sampling is $\delta$-noisy, where the value of $\delta$ is the noise level provided by the acquisition system. We make no assumption about the noise model distribution, and we apply no denoising or filtering algorithm to minimize noise effect.

### 2.2.2 Sample density

An important measure for analyzing sampled surfaces is the sample density, which is the surface area associated with each sampled point. Intrinsically, the sample density specifies the minimum feature size, or level of details, of the shape captured around a point. The density estimation on a continuous surface can be computed as [DHS00]:

$$Pd = \int_{\mathcal{R}} \rho(\mathbf{x}) d\mathbf{x}, \tag{2.4}$$

where $Pd$ is the probability that a point $\mathbf{x}$ will fall in the region $\mathcal{R}$ and $\rho(\mathbf{x})$ is the continuous density function, averaged by $Pd$. If we assume that the region $\mathcal{R}$ is so small that $\rho$ does not vary appreciably within and the density function $\rho(\mathbf{x})$ at a point $\mathbf{x}_i$ can be approximated by $\hat{\rho}(\mathbf{x}_i)$:

$$\hat{\rho}(\mathbf{x}_i) = \frac{k/n}{V_{\mathcal{R}}} \tag{2.5}$$

where $V_{\mathcal{R}} \in \mathbb{R}^3$ is the infinitesimal volume enclosed by $\mathcal{R}$, $\mathbf{x}_i$ is a point within $\mathcal{R}$, $k$ is the number of samples inside $\mathcal{R}$ and $n$ is the total number of samples.

From 2-manifold surface assumption, we have that on some neighborhood $\mathcal{N}(\mathbf{x}_i)$ around $\mathbf{x}_i$, $\mathcal{N}(\mathbf{x}_i)$ lies on a disk. Let $r_i$ be the disk radius and $|\mathcal{N}(\mathbf{x}_i)|$ be the number of samples inside the neighborhood $\mathcal{N}(\mathbf{x}_i)$. The local density (2.5) around a point $\mathbf{x}_i$ is then given by:

$$\hat{\rho}(\mathbf{x}_i) = \frac{|\mathcal{N}(\mathbf{x}_i)|}{\pi r_i^2} \tag{2.6}$$

The local sampling density, given by (2.6), gives the number of sample per unit area. The main drawback of using (2.6) to estimate the local sampling density is that the neighborhood at a each point must be provided. Instead of using this measure, we will use the local sampling spacing. The local spacing, $\eta(\mathbf{x}_i)$, measures the average distance between points within the disk defined by the neighborhood $\mathcal{N}(\mathbf{x}_i)$. From the local density estimation, we can derive the local sampling spacing as:

$$\eta(\mathbf{x}_i) = \frac{1}{\sqrt{\hat{\rho}(\mathbf{x}_i)}}. \tag{2.7}$$

An alternative estimation of the sampling density is the minimum sampling spacing, $\eta_{\min}(\mathbf{x}_i)$, which is the distance from the point $\mathbf{x}_i$ to its closest neighbor:

$$\eta_{\min}(\mathbf{x}_i) = d(\mathbf{x}_i, X) \tag{2.8}$$

where

$$d(\mathbf{x}_i, X) = \inf_{\mathbf{x}_j \in X} \{\|\mathbf{x}_j - \mathbf{x}_i\|\} \tag{2.9}$$

We say a point cloud $X$ is $\varepsilon$-dense if the minimum local sampling spacing $\eta_{\min}(\mathbf{x}_i) < \varepsilon$ holds true for any $\mathbf{x}_i \in X$. This definition of $\varepsilon$-dense imposes the sample to be dense anywhere on $\Sigma$ and it gives a lower bound to the density. The minimum local sampling spacing is the measure most commonly used to estimate the density of a sample. This is due to the simplicity it is

computed, since it does not require the extraction of the neighborhood around a point.

### 2.2.3 Holes, outliers and feature size

The point cloud $X$ is assumed to be near an unknown surface $\Sigma$. Because of the sampling process, it is observed in $X$ the presence of artifacts, such as holes, outliers, and noise. Noise samples and outliers do not lie exactly on the surface $\Sigma$, and they modify locally the shape of the discrete surface representation. Noise is hard to be identified, because its high frequency profile is ambiguous with sharp features. Holes, or missing data can be interpreted as holes in $\Sigma$. Therefore, the sampling model needs to specify the dependence of the approximation on these artifacts.

Intuitively, for noiseless, dense, uniform samplings, samples do not leave holes of radius larger than $\varepsilon$. Regions without samples that are bigger than $\varepsilon$ are interpreted as holes in $\Sigma$. If the sample is $\delta$-noisy, holes cannot have a radius larger than $\delta+\varepsilon$. In the case where these conditions are not respected, which is commonly found in point clouds obtained from range sensors, holes cannot be recovered and these regions are viewed as surface holes.

Another intrinsic propertiy of the surface $\Sigma$ is its features, which encompass local geometry and topology. The minimum feature size captured by the samples can also be inferred from $\varepsilon$ and $\delta$. For noiseless and uniform sampling, features on $\Sigma$ smaller to both either $\varepsilon$ and $\delta$, are not recoverable.

Assuming a $\varepsilon$-dense, $\delta$-noisy point cloud, outliers can be detected by looking at $\eta_{\min}(\mathbf{x}_i)$ for each $\mathbf{x}_i$. For $\delta = 0$, we can deduce that points $\mathbf{x}_i$ with $\eta_{\min}(\mathbf{x}_i) > \varepsilon$ cannot be a point of $\Sigma$, since that would violate the condition $\varepsilon$-dense for $X$. In such scenario, $\mathbf{x}_i$ is taken as an outlier. For a $\varepsilon$-dense, $\delta$-noisy, a point $\mathbf{x}_i$ is an outlier if $d(\mathbf{x}_i, X) > \varepsilon + \delta$.

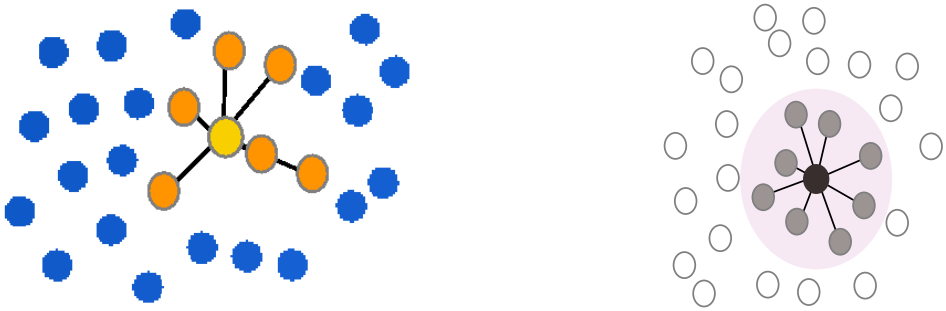
## 2.3 Structuring the point clouds

In section 2.2, we presented the definition of 2-manifold surface, where surface properties are defined on local neighborhood $\mathcal{N}$, for $\mathcal{N} \to 0$. In this section we present how we compute the neighborhood $\mathcal{N}$ for the point cloud. We consider two neighborhood systems: the $k$-nearest and the ball neighborhoods. Using points connectivity established through neighborhood information, we construct a geometric graph called *neighborhood graph*, which gives a discrete representation of the sampled surface.

### 2.3.1 Neighborhood systems

A neighborhood system determines points connectivity through spatial relationship between the points. The neighborhood of a point $\mathbf{x}_i$ is a subset $\mathcal{N}(\mathbf{x}_i) \subset X$ that contains $\mathbf{x}_i$, in which all points from this set satisfy a certain neighborhood condition and adequately represent a small surface patch around $\mathbf{x}_i$.

The neighborhood of a point $\mathbf{x}_i$ only depends on the geometric locations of the points in space, not on some additional combinatorial structure associated with the point cloud [Pau03]. The various differential properties of surfaces presented in this thesis will be defined on one of



(a) (b)

**Figure 2.2:** *Local neighborhood systems: (a) k-nearest neighborhood, (b) ball neighborhood.*

these neighborhoods. We will describe the $k$-nearest and the ball neighborhood system, and their principle is illustrated in Figure 2.2.

### k-nearest neighborhood

The $k$-nearest neighborhood of a point $\mathbf{x}_i \in X$ consists of the $k$ closest points to $\mathbf{x}_i$ in $X$. Let $\Omega$ be a permutation, or an ordered list, where $\|\mathbf{x}_{\Omega(i,1)} - \mathbf{x}_i\| > 0$ and $\|\mathbf{x}_{\Omega(i,j)} - \mathbf{x}_i\| \leq \|\mathbf{x}_{\Omega(i,j+1)} - \mathbf{x}_i\|$, for $j \in [1, n-1]$. This permutation gives the ordering of all points in $X$ according to their Euclidean distance to the point $\mathbf{x}_i$. The $k$-nearest neighborhood of the point $\mathbf{x}_i$ is given by

$$\mathcal{N}_k(\mathbf{x}_i) = \{\mathbf{x}_{\Omega(i,1)}, \cdots, \mathbf{x}_{\Omega(i,k)}\}. \tag{2.10}$$

As the number of points $k$ is fixed, the method adapts the neighborhood size according to the point density. The set $\mathcal{N}_k(\mathbf{x}_i)$ defines the neighborhood size as a sphere centered in $\mathbf{x}_i$ with radius $r_i = \|\mathbf{x}_{\Omega(i,k)} - \mathbf{x}_i\|$.

The $k$-nearest neighborhood system is preferred over most reconstruction algorithms [HDD+94, DLS05, ZPKG02]. This system has the advantage to adapt the neighborhood size to local changes in sampling density. However, when working directly on raw points, this neighborhood system does not guarantee that points of $\mathcal{N}_k(\mathbf{x}_i)$ represent a small, local surface patch around the point. Spurious and sparse data are incorrectly connected to surface points, as showed in Figure fig. 2.3.

### Ball neighborhood

The ball neighborhood of a point $\mathbf{x}_i$ consists of all points in $X$ that are inside a sphere centered in $\mathbf{x}_i$ with a radius $r_i$, denoted $\mathcal{N}_{r_i}(\mathbf{x}_i)$:

$$\mathcal{N}_{r_i}(\mathbf{x}_i) = \{\mathbf{x}_j \in X \mid \|\mathbf{x}_j - \mathbf{x}_i\| < r_i\}. \tag{2.11}$$

The sphere radius $r_i$ can be data driven, and it can adapt to sampling variation and local shape. The neighborhood system based on an unique sphere radius is preferable when one wants to have control of the area of interest, with a neighborhood size dependent on local or global

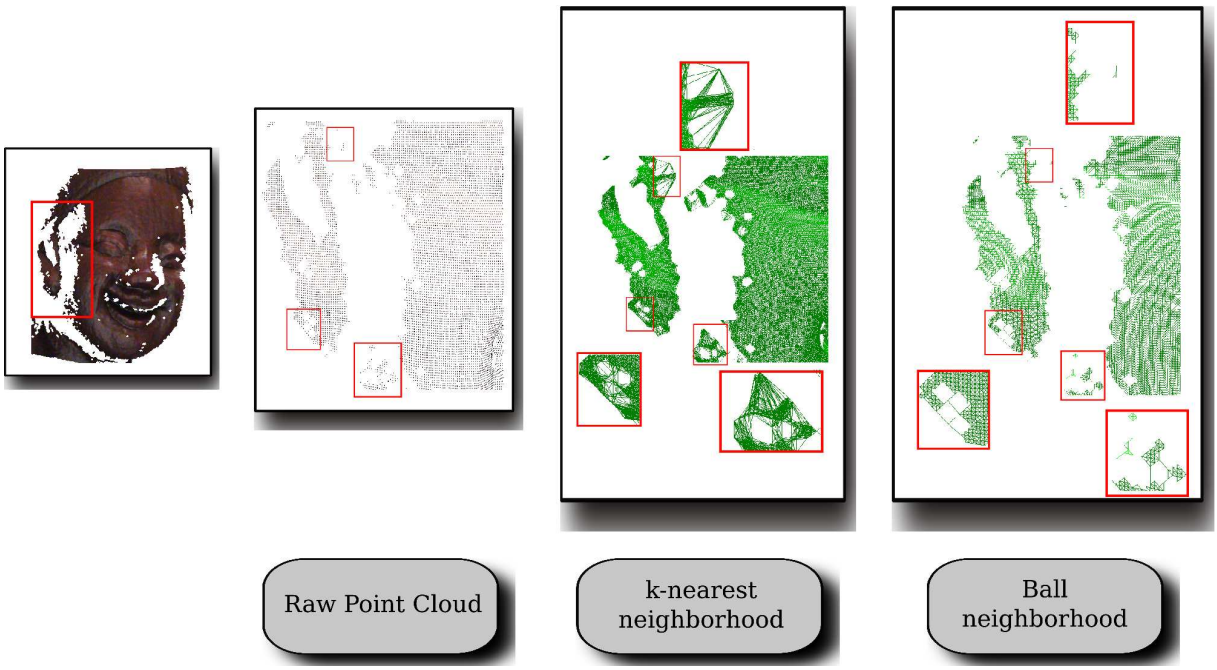



**Figure 2.3:** *Effect of neighborhood system choice. Observe that the k-nearest neighborhood system connects points with outliers and regions separated by acquisition holes. Such a behaviour is not observed for the ball neighborhood.*

sampling density. Additionally, this last choice allows the identification of spurious and sparse data.

### 2.3.2 Structuring point cloud through the *neighborhood graph*

We use a *neighborhood graph*, which is a generalization of the two most used geometric graphs to process point clouds: ball graphs, and $k$-nearest neighbors graphs. In this section we present the definition of the *neighborhood graph* and its main properties.

#### *Neighborhood graph* definition

A *neighborhood graph* $G_{\mathcal{N}}$ is represented by an undirected weighted graph

$$G_{\mathcal{N}} = (V, E, W) \tag{2.12}$$

where the points $\mathbf{x}_i \in X$ (2.2) form the node set $V$. The edge set $E$ is given by

$$E = \{(\mathbf{x}_i, \mathbf{x}_j)\} \subset X \times X \tag{2.13}$$

A mapping defined on $E$ uniquely assigns to every element of $E$ a non-ordered pair of distinct elements of $V$. The pair $(\mathbf{x}_i, \mathbf{x}_j)$ is in $E$ only if the neighborhood predicate $\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)$ is verified. Associated with each edge $(\mathbf{x}_i, \mathbf{x}_j) \in E$ we have the weight $w_{ij}$, which assigns a non-negative similarity measure $w : E \to \mathbb{R}^+$. Examples of weighting function are the Euclidean distance in some feature space, the angle between vectors, or a Gaussian distribution. The weight set is

given by

$$W = \{w_{ij}\} \subset \mathbb{R}^+ \tag{2.14}$$

The *neighborhood graph* (2.12) is defined by

$$G_{\mathcal{N}} = (\{\mathbf{x}_i\}, \{(\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)\}, \{w_{i,j}\}) \tag{2.15}$$

As a geometric graph, the *neighborhood graph* can be efficiently constructed, thanks to the many data structures (KD-trees, voxel grids, ball trees) available for range internal queries search methods that gives nearest neighbor results in near-linear time.

Given a *neighborhood graph* $G_{\mathcal{N}}$ structuring a point cloud, missing data, boarder points, noise presented in $X$ can generate disjoint sub-graphs, with variable node connectivity and isolated points. Even when the neighborhood size is data-driven and adaptative, graph connectivity is not regular and it tends to form too many edges between points located in high dense regions, and less connections are observed in regions where points are sparse. Figure 2.3 illustrates such situation.

#### *Neighborhood graph* properties

We briefly describe some concepts from basic graph theory used here to characterize a *neighborhood graph*. An edge sequence, $\tau = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, is called a *path* if $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ are pairwise distinct vertices. The length of a path $\tau$ is defined as $s(\tau)$, and it is the sum the weights of of all edges $(\mathbf{x}_i, \mathbf{x}_j) \in \tau$:

$$s(\tau) = \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \tau} w_{i,j} \tag{2.16}$$

If a *path* between two points $\mathbf{x}_1$ and $\mathbf{x}_n$ does not exist, we have $s(\tau) = \infty$. An example of weighting function is $w_{ij} = \{(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)\}^{\frac{1}{2}}$ and we have an Euclidean graph, in which the *path* distance between two points with the smallest length is taken to approximate the geodesic distance. A *cut* in a graph is the edge set that, when removed, promote the partition of the nodes into two disjoint sets. A *crossing edge* is any edge from the *cut*.

A spanning tree $S_G$ of a connected graph $G$ is a sub-graph containing all nodes of $G$ in which there is an unique *path* between any two points $\mathbf{x}_i, \mathbf{x}_j \in S_G$. The minimum spanning tree ($MST$), $\Gamma_G$, of a connected graph $G$ is, from all possible spanning trees, the one connecting all nodes of $G$, where the sum of all edges is minimum. Since the *neighborhood graph* $G_{\mathcal{N}}$ is not necessarily connected, each connected sub-graph, $G'_{\mathcal{N}} \subset G_{\mathcal{N}}$, has one corresponding minimum spanning tree, $\Gamma_{G'_{\mathcal{N}}}$, which isn't necessarily unique [Sed02, DHS00]. Therefore, the *neighborhood graph* $G_{\mathcal{N}}$ is represented by a set of disjoint minimum spanning trees, or a minimum spanning forest, $\{\Gamma_{G'_{\mathcal{N}}}\}$.

### 2.3.3 Implementation

The central computational primitive required by the *neighborhood graph* construction is the closest points query: given a point $\mathbf{x}_i \in X$ find its neighbors $\mathcal{N}(\mathbf{x}_i) \subset X$. There are many proposed efficient data structures and search methods (kdtree [Ben75], octree, etc) to find efficiently the

neighborhoods of a point. We use a hash data structure, where the coordinates of arbitrary points are mapped to a cell [TW04]. The cell size is fixed and proportional to the neighborhood size. To find the neighborhood of a point, we search on the cell where the point is inserted, and, if necessary, on the adjacent cells. The insertion of a point can be done in $O(1)$, and the querying takes $O(q)$, where $q$ is the maximum number of points in a cell. In practice, with a sufficient number of cells $q$ will be small. We store the neighborhood information on a graph list, where we can access directly the neighbors of any point. The edges can be weighted in any desired space.

A second important procedure is the extraction the $MST$. The $MST$ is a sub-graph of the *neighborhood graph*, and we use the Prim's algorithm [Pri57]. The Prim's algorithm, as other $MST$ construction algorithms, is based on the following two properties [Sed02]:

1. *Identifying edges that must be in a MST:* given any *cut* in a graph $G$ (fig. 2.4, top left), the minimal crossing edge (fig. 2.4, top right) of such *cut* belongs to some $MST$ of the graph.

2. *Identifying edges that must not be in a MST:* given a graph $G$ (fig. 2.4 top left), if we add a new edge $(\mathbf{x}_i, \mathbf{x}_j)$, the new $MST$ is the one constructed by adding $(\mathbf{x}_i, \mathbf{x}_j)$ to the original $MST$ and deleting the edge with biggest weight on the resulting cycle (fig. 2.4 bottom left).

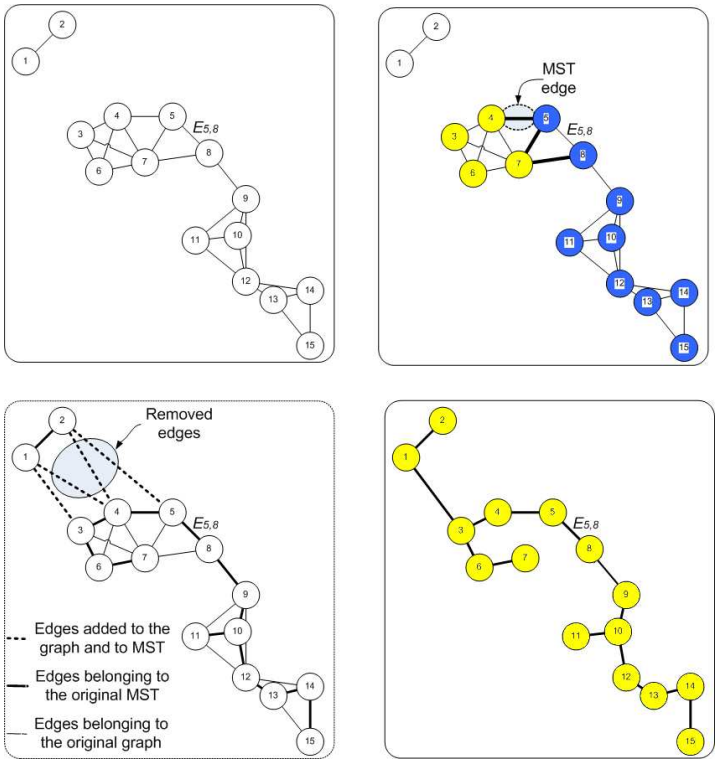



**Figure 2.4:** *Minimum spanning tree properties.*

Prim's algorithm traverses the nodes of the *neighborhood graph* and, for each node, it adds to the $MST$ the edge $(\mathbf{x}_i, \mathbf{x}_j) \in E$ with minimal *crossing edge*. The edge $(\mathbf{x}_i, \mathbf{x}_j)$ must not

generate a cycle on the resulting $MST$. When representing a graph $G$ by a adjacency list, the Prim's algorithm runs in time $O(|E|log|V|)$, which consists in a traversing the graph $G$.

## 2.4 Extracting geometric features from point clouds

If the point set $X$ sampled over an inferred surface $\Sigma$ is all we know about $\Sigma$, it is impossible to recover the shape, or extract higher level information, of $\Sigma$ from $X$. Some additional knowledge of the underlying shape represented by the point cloud is required. There are several local surfaces properties that can be estimated from the point cloud $X$ that provide information about the local shape around the point [PWHY09, JH97, FHK+04]. In this section we will examine two methods to compute local shape descriptors, which provide curvature information. They can be efficiently estimated on large raw point clouds. These methods, however, assume that an oriented normal vector at each point is provided. This we will also examine a method to estimate efficiently and robustly the normal vector operating directly on the noisy point cloud. The main challenge of operating directly on raw point clouds, to compute the normals and the curvature at each point, is to provide estimations that are robust to noise and to re-sampling.

Shape is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object [DM98]. A descriptor vector $f(\mathbf{x}_i)$ can be both estimated from the input data or provides by the acquisition system. It must be invariant to coordinate transformation and robust to noise, so it can be used in the registration pipeline. Examples of local descriptors are curvature, normal cone, color, material properties and textures coordinates. These descriptors can be extracted separately and they can be combined to describe the sampled surface $X$. Several local descriptors can be used to describe the point $\mathbf{x}_i \in X$. The descriptor vector $f$ is defined as a set of features which characterizes the point, and is given by:

$$f: \quad \mathbf{x} \quad \in X \rightarrow \mathbb{R}^m \tag{2.17}$$

$$\mathbf{x}_i \quad \rightarrow f(\mathbf{x}_i) \tag{2.18}$$

Finally, we have the descriptors set $I$, defined as

$$I = \{f(\mathbf{x}_i)\} \subset \mathbb{R}^m \tag{2.19}$$

which together with the point cloud $X$, form a representation of the surface $\Sigma$. The point cloud is then characterized by points location, in Euclidean space, $\mathbb{R}^3$, and in descriptors space, $\mathbb{R}^m$. In this thesis, we will restrict our attention to curvature-based, low-dimensional, descriptors, since they are cheaper to compute, store, and compare. In the following section, we will show how we estimate two local shape descriptors directy from point cloud: the normal vector (Section 2.5) and the principal curvatures (Section 2.6).

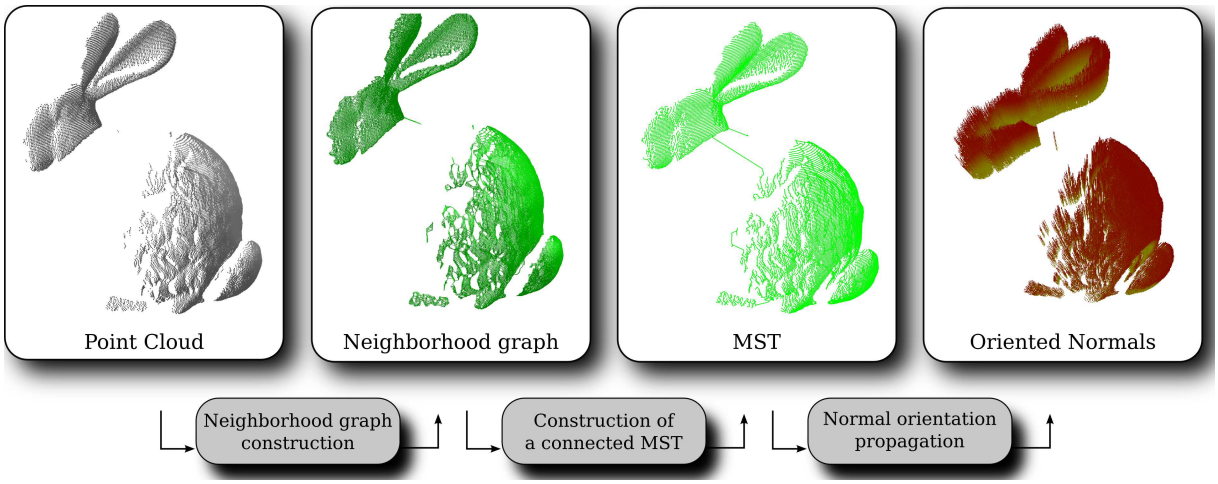



**Figure 2.5:** *Normal vector estimation pipeline using Hoppe-Johnson's schemes.*

## 2.5 Normal estimation

Most algorithms used to estimate local shape descriptors assume the knowledge of surface orientation at each point. In this section, we formulate the normal estimation problem as a plane fitting problem, using [HDD+94], and we propose an algorithm to orient these normals consistently directly from the raw point cloud.

### 2.5.1 Related works on normal estimation

There are several approaches in the literature that proposed to compute the normals directly from an unorganized point cloud and they are classified into non-parametric [AB98, TM98, TTMM04] or parametric [LJLC05, ABCO+03, ZPKG02] approaches.

Amenta *et al.* [AB98] presented the CRUST algorithm, a non-parametric normal estimation approach. Given a Voronoi diagram structuring a point cloud $X$, they have showed that the normals can be estimated by the poles of the Voronoi diagram, which are the furthest Voronoi vertices from the respective sites on the two sides of the sampled surface. In the noise-free case, the normals estimated using the CRUST algorithm agree with the actual normals, but the same property does not hold true for noisy point cloud. Dey *et al.* in [DG04, TKD05] extended the work of Amenta, and they presented the COCONE algorithm, also a Voronoi/Delaunay based method for estimating normals from noisy point cloud data. The main drawback of Voronoi-based methods is that they make the restrictive assumption of a closed bounded shape, necessary to access the medial axis, in order to estimate the normals. Unfortunately, the medial axis is not known for most sampled surfaces, e.g., scanned models.

Most practical normals estimation algorithms are parametric approaches, where the normals estimation is seeing as a local geometric fitting problem. The parametric methods for local surface analysis are based on the assumption that the $k$-nearest neighbors of a sample point adequately represent a small patch of the underlying surface [Pau03].

In [LJLC05], an arc-length parameterized third-order approximation is fit to a small neighborhood at a point. The coefficients of this model are computed by solving a weighted-least

squares problem at each point using only the points in its local neighborhood. This procedure estimates the curvature and torsion, as well as the tangent as a by-product. Cazals *et al.*[CP03] fit the local representation of a manifold using coefficients of a truncated Taylor expansion, termed a jet, to a small neighborhood around a point.

Hoppe *et al.* in [HDD+92, HDD+94] estimate the normal at each point by fitting a least square plane to its $k$-nearest neighbors. The neighborhood size in [HDD+92] is fixed through visual inspection. The main advantage of this approach is its robustness to noise due to its filtering effect and its main drawback is that the accuracy of the estimate is highly dependent on neighborhood size. The plane model used by Hoppe is simple in form and has been used by other researchers for denoising [MN03], reconstruction [ABCO+03, HDD+92, ZPKG02], and registration tasks.

The neighborhood size used to estimate the normal determines the scale of analysis and, consequently, the accuracy of the estimation and the robustness to noise. Many works have been proposed to find the neighborhood size that best fit the data to the model. Mitra in [MN03], Unnikrishnan in [ULVH06b, Unn08] and Pauly in [PKKG03], for example, studied the effect of neighborhood size, curvature, sampling density, and noise on the Hoppe's normal estimation technique. Mitra in [MN03] analyses the scatter matrix, formed from the neighborhood of a point $\mathbf{x}$, to see how curvature, noise, neighborhood size, and sampling density affects this matrix and consequently, the normal estimation. Instead of using the $k$-neighborhood system, they take a ball neighborhood of radius $r_i$, estimated for each point $\mathbf{x}_i$. Under the assumption that the point cloud noise has zero mean and standard deviation $\sigma_n$, the curvature $\kappa_i$ is constant, and the samples are evenly distributed (satisfy the $(\epsilon, \delta)$ sampling conditions proposed by Dey in [Dey06]), the neighborhood size that minimizes the angle between the estimated normal and the true normal is given as [MN03]:

$$r_i = \left( \frac{1}{\kappa_i} \left( d_1 \frac{\sigma_n}{\sqrt{\epsilon \rho}} + d_2 \sigma_n^2 \right) \right)^{1/3} \tag{2.20}$$

for some constants $d_1, d_2$. Mitra suggests an iterative scheme in which it is estimated the local density, the local curvature and the neighborhood size (2.20). The main drawback of the closed form expression (2.20) is that it involves two parameters that rely on the knowledge of the observed data distribution and they have to be fixed *a priori*. A similar approach was proposed by Unnikrishnan in [ULVH06b, Unn08].

The performance of different methods to estimate normals directly from noisy point clouds has been examined in [DG04]. In [DG04], Dey compared the performance of Voronoi-based methods with least square plane fitting methods in [MN03, PKKG03]. The first important observation is that all methods performed equally well for noisy datasets and non uniform sampling. As the size of the input point cloud increases, they have showed that the plane fitting algorithms [MN03, PKKG03] should be preferred over [DG04], once Voronoi-based methods are time and memory consuming. These considerations have motivated us to use a least square plane fitting method to approximate normals from raw point cloud, and we will detail such an algorithm in the next section.

### 2.5.2 Normal estimation using Hoppe's method

We consider the problem of approximating the normal for a point cloud in $\mathbb{R}^3$ and we use the algorithm proposed by Hoppe *et al.* [HDD+92] in surface reconstruction context. In [HDD+92], the normal vector is estimated through principal component analysis (PCA). The normal vector estimation is formulated as a local geometric fitting problem. In the process of doing so, it is assumed that a subset of observed points in a small neighborhood $\mathcal{N}_{r_i}(\mathbf{x}_i)$ is homeomorphic to a disk. If the surface is smooth, it is reasonable to expect that the scatter matrix will be elongated and that its major axis, or principal eigenvector, will approximate the direction of the plane for some appropriate neighborhood size. Such situation is illustrated in Figure 2.6.

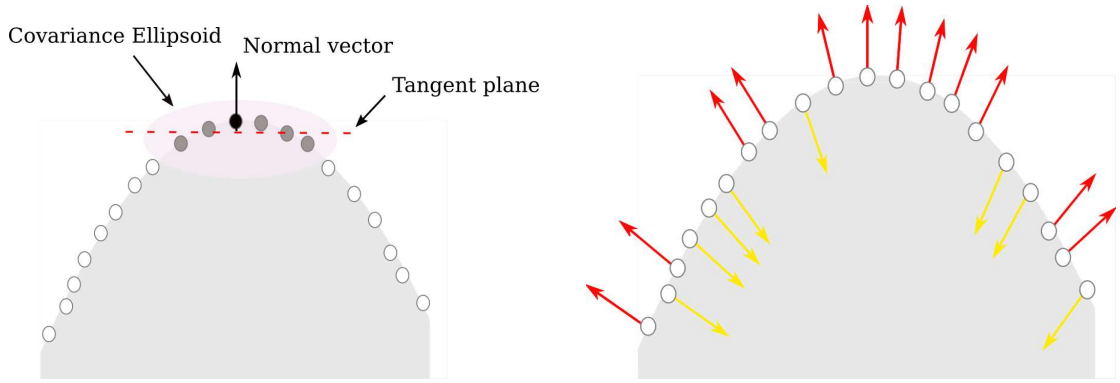



**Figure 2.6:** *Normal estimation. (a) Estimating the tangent plane using covariance matrix. (b) Normal orientation, where the red normals should be flipped, since they are not pointing outward.*

Let $\mathcal{N}_{r_i}(\mathbf{x}_i)$ be the neighborhood associated with the point $\mathbf{x}_i$. The covariance matrix in this neighborhood is given by:

$$\mathbf{M} = \sum_{\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)} (\mathbf{x}_j - \overline{\mathbf{x}_i})(\mathbf{x}_j - \overline{\mathbf{x}_i})^T \tag{2.21}$$

where $\overline{\mathbf{x}_i}$ is the centroid of $\mathcal{N}_{r_i}(\mathbf{x}_i)$, given by

$$\overline{\mathbf{x}_i} = \frac{1}{|\mathcal{N}_{r_i}(\mathbf{x}_i)|} \sum_{\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)} \mathbf{x}_j \tag{2.22}$$

Note that $\mathbf{M}$ is a symmetric $3 \times 3$ positive matrix. It describes the statistical properties of the distribution of the samples in the neighborhood $\mathcal{N}_{r_i}(\mathbf{x}_i)$. The diagonal elements of $\mathbf{M}$ are the variances and the off-diagonal elements are the covariances. The eigenvector problem is stated as:

$$\mathbf{M}\mathbf{v}^l = \lambda^l \mathbf{v}^l, l \in \{1, 2, 3\} \tag{2.23}$$

From PCA (2.23), we have that the eigenvectors of a real and symmetric scatter matrix are orthogonal and form the principal axes that best fit a plane to the samples, in least-sum-of-squared-error sense [DHS00]. The eigenvalue $\lambda_i^l$ measures the variation of the points along the

direction of $\mathbf{v}_i^l$. If $\lambda_i^1 \leq \lambda_i^2 \leq \lambda_i^3$ correspond to the eigenvalues of $\mathbf{M}$ associated with the unit eigenvectors $\mathbf{v}_i^1$, $\mathbf{v}_i^2$, $\mathbf{v}_i^3$, respectively, the normal vector $\hat{\mathbf{n}}_i$ is the eigenvector associated with the smallest eigenvalue.

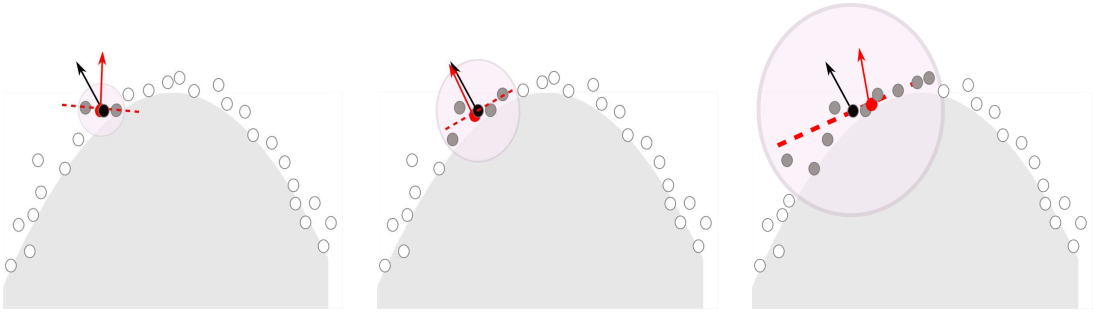
A way to evaluate the tangent plane estimation is by looking at the points dispersion along the tangent plane. Let $\lambda_i^1$ be the smallest eigenvalue, an estimate of the normal deviation, $\sigma_{\hat{\mathbf{n}}_i}$, in $\mathcal{N}_{r_i}(\mathbf{x}_i)$, is given by [ZPKG02]

$$\sigma_{\hat{\mathbf{n}}_i} = \frac{\lambda_i^1}{\lambda_i^1 + \lambda_i^2 + \lambda_i^3} \tag{2.24}$$

For a noiseless flat surface, $\sigma_{\hat{\mathbf{n}}_i} = 0$. The maximum surface variation $\sigma_{\hat{\mathbf{n}}_i} = 1/3$ happens when a completely isotropically distribution of points is observed [PGK02].

In this formulation, it is assumed that the curvature $\kappa$ of the underlying manifold is bounded and near constant, i.e., $\dot{\kappa} = 0$, and the estimate is only accurate when this hypothesis is hold true. Artifacts usually observed in raw point clouds, such as local sampling density variation, boundary, holes, and outliers can bias the centroid of the estimated plane out of $\mathbf{x}_i$. Such behaviour results in the estimation of a tangent plane that does not lie at $\mathbf{x}_i$. All these artifacts affect the accuracy of the normal vector estimation, but their removal is still an ill-conditioning problem [UH07].

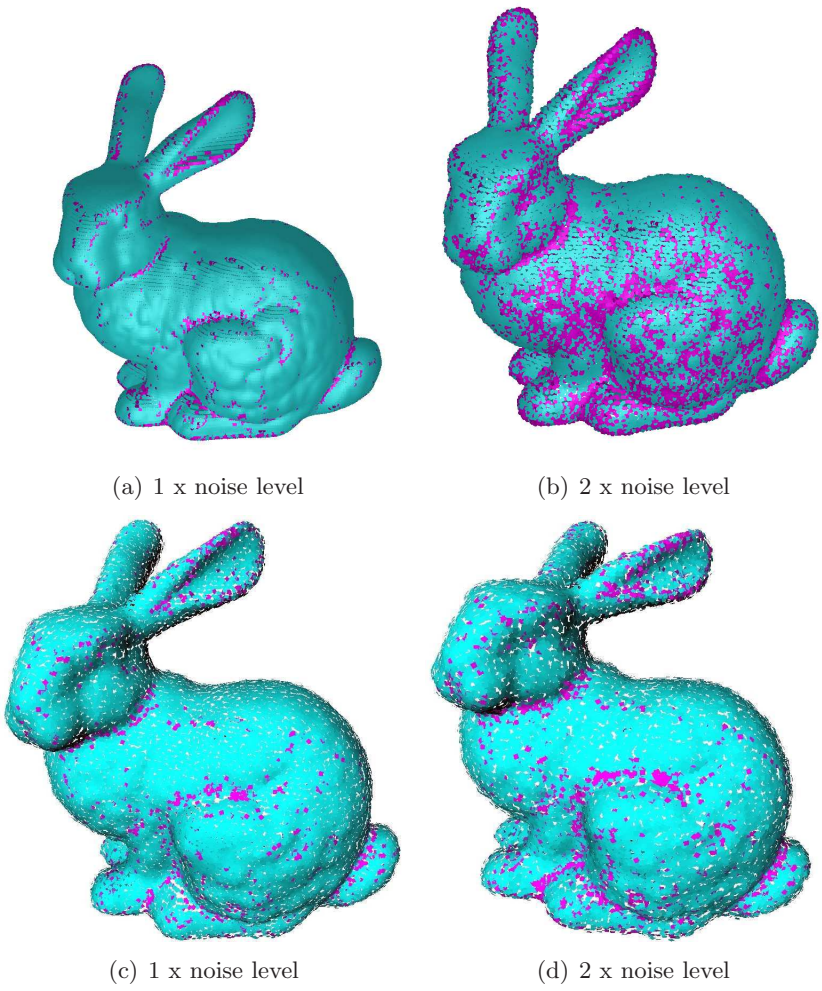
The only input parameter of the algorithm is the neighborhood size. It determines the scale selection at point and, consequently, the magnitude of which samples noise, distribution, and curvature affect the normal estimation. Figure 2.7 illustrates the neighborhood influence on the estimate. Small neighborhood size (Figure 2.7 (a)) can compromise the quality of the estimate due to the use of small number of noisy data points, while large neighborhood (Figure 2.7 (c)) the surface patch considered is distorted and it is no longer homeomorphic to a disk.



(a) (b) (c)

**Figure 2.7:** *Effect of neighborhood size on normal estimation. The expected normal vector is showed in black. The tangent plane, the centroid, and the normal estimated are illustrated in red.*

In the present work, we use a unique neighborhood size, where the size of the neighborhood radius is estimated proportional to the average local sampling spacing over the samples. Alternatively, Mitra in [MN03] and Unnikrishnan in [ULVH06b] have addressed the problem of improving the normal estimation accuracy through an adaptive neighborhood size. In Figure 2.8 we illustrate the influence of the neighborhood on the estimation accuracy. We compare the normal estimation error for a fixed neighborhood size (fig. 2.8 (a),(b)) and an adaptive



(a) 1 x noise level (b) 2 x noise level

(c) 1 x noise level (d) 2 x noise level

**Figure 2.8:** *Effect of using an adaptive or a fixed neighborhood size to estimate the normals. (a) and (b): normals estimated using a fixed* 10*-nearest neighborhood. (c) and (d): normals estimated using an adaptive neighborhood size [MN03]. It was added a Gaussian noise, where the magnitude of the added noise was measured in a scale where the average spacing between neighboring points in the mesh representation of the original data was taken as one unit. It is highlighted the points with estimation error more than* 5 *degrees under two different amounts of noise.*

neighborhood size (fig. 2.8 (c),(d)), using Mitra's algorithm.

For both noise levels, the adaptive neighborhood size method works better than fixed neighborhood size method. However, for low noise levels, which are close to the noise found in raw point clouds, the error of normal estimation are similar. We point out that the plane fitting approach leads to smooth sharp features even when an adaptive neighborhood is used. This is because the sampling requirement to feature representation [Dey06] on high frequency features is not respected and, consequently, there is no possible neighborhood homeomorphic to a plane. The main drawback of using the same neighborhood size over the entire point cloud is to find the scale with best tradeoff between being robust to noise and accurate at sharp features. When estimating the normals of large raw point clouds, the latest approach is preferred because of its efficiency.

### 2.5.3 Normal orientation

A consistent normal orientation can be computed using a method based on the Euclidean minimum spanning tree of the point cloud, as described in [HDD+92]. For raw point clouds, where the object topology is unknown, a global consistent normal orientation is still a ill-conditioning problem. A global coherent orientation is expected, even when missing data produce not connected regions. We try to solve this problem by using both Hoppe's orientation propagation approach [HDD+92] and Jonhson's global orientation heuristic [Joh97].

Hoppe's orientation propagation algorithm is based on the assumption that the directions of the normals at nearby points on $\Sigma$ cannot vary abruptly. In other words, it is based on the assumption that the surface looks flat locally. Let two points $\mathbf{x}_i$ and $\mathbf{x}_j$ be geometrically close, the point cloud be globally dense and the local curvature be bounded. In this configuration, the tangent planes formed by $\mathcal{N}_{r_i}(\mathbf{x}_i)$ and $\mathcal{N}_{r_j}(\mathbf{x}_j)$ are parallel [HDD+92]. Therefore, $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j \approx \pm 1$, and $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j \approx +1$ if they are oriented in the same direction. When this condition is verified on every point pair connected by an edge of the *neighborhood graph*, the normals orientation are globally coherent.

Points connectivity on *neighborhood graph* are highly redundant and the order in which the orientation is propagated becomes an important issue. Hoppe solved the problem of finding the order in which the normal vector is propagated as a graph optimization problem. Let $G_{\mathcal{N}}$ be an Euclidean *neighborhood graph*. It is constructed a sub-graph by traversing the *minimal spanning tree* ($MST$) of the resulting graph. This choice was made because in the $MST$, the shortest path is always preferred [Sed02]. The final sub-graph transverses the object passing by regions of low curvatures and skirting sharp edges.

This method produces a correct normal orientation for smooth, connected point clouds. However, none of neighborhood systems guarantee that only one connected Euclidean graph represents the raw point cloud. Thus, the *neighborhood graph* can be composed by a set of non-connected sub-graphs. In this situation, often observed on scanned surfaces, the global normal orientation frame presented in [HDD+92] does not give geometric constraints that allow the propagation of the normal vector orientation from one disjoint part to another. During the construction of the $MST$ sub-graphs, disjoint parts were recognized and marked, so they could be treated separately in the orientation propagation phase.

Rooting the $MST$ at the initial node, we traverse the tree in depth-first order, spreading the normal in this order. The goal of this action is to coherently orient the normal vectors of connected sub-graph. Discontinuity is respected. The first point of each disjoint part is considered as a reference point. Let $(\mathbf{x}_i, \mathbf{x}_j)$ be an edge, with $\mathbf{x}_i$ being their parent node. It is assumed the parent node has the correct normal direction. The cost assigned to the connection is $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j$. The orientation of $\hat{\mathbf{n}}_j$ is inverted if the vectors have opposite directions, or, in other words, if $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j < 0$.

Once all normal vectors are properly oriented, it is necessary to verify the correctness of the normal sense, which is desired to be oriented to outside of the object. The algorithm used is the heuristic one proposed by Johnson in [Joh97]. In [Joh97], the correctness of global normal orientation is determined by calculating the dot product of the normal vector at each node and

the vector from the centroid to the point:

$$O_i = \sum_{\mathbf{x}_i \in X} Op_i - \sum_{\mathbf{x}_i \in X} On_i \tag{2.25}$$

where,

$$Op_i = \begin{cases} 1 & \text{if } \hat{\mathbf{n}}_i \cdot (\mathbf{x}_i - \mathbf{o}_i) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.26}$$

and

$$On_i = \begin{cases} 1 & \text{if } \hat{\mathbf{n}}_i \cdot (\mathbf{x}_i - \mathbf{o}_i) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.27}$$

where $\mathbf{o}_i$ is the centroid of the tangent plane perpendicular to $\hat{\mathbf{n}}_i$. Equation (2.25) analyzes the frequency that the samples and the normals are on the same side of the tangent plane. It is assumed that, if $O_i > 0$, the normals have been oriented to the outside. Otherwise, the orientation of the set points should be inverted. This procedure is repeated for each sub-graph separately.

The orientation algorithm presented here was used in different examples with different resolutions, complexity and level of discontinuity, and it has produced correct orientation in most of the cases we have run. However, for more complex surfaces and disconnected sub-graphs, the algorithm do not always produce correct global normal orientation. Such behavior deteriorates the point cloud visualization and the *principal curvatures* estimation. While visualization uses normals orientation to visibility and lighting computation, *principal curvatures* estimation algorithm uses normals orientation to determine local shape concavity.

### 2.5.4 Evaluation of the performance of the normal estimation algorithm

In this section, we evaluate the normal estimation algorithm performance, presented in section 2.5, on raw unstructured point cloud. Particularly, we analyze the accuracy of the normal estimation when the following aspects are verified:

- The input data presents a moderate noise level.
- The 2-manifold hypothesis is not verified.
- The stability of estimation when local sample density slightly changes.

The angle between the reference vector and the estimated vector is the error metric adopted, and it is expressed in degrees or radians. Obviously the smaller the error is, the better the estimation is. The reference normal for a point $\mathbf{x}_i$ is computed as the average of the normals of the triangles incident to a vertex $\mathbf{x}_i$ in the mesh representation of the surface. To compute the reference normals, we need the data to have a triangle mesh representation. We obtain a surface representation for a given a unstructured point cloud by using surface reconstruction software, such as COCONE [COC] and 3DReshaper [3DR]. When the ball neighborhood system is chosen, the neighborhood size parameter is taken proportional to the average sampling spacing.

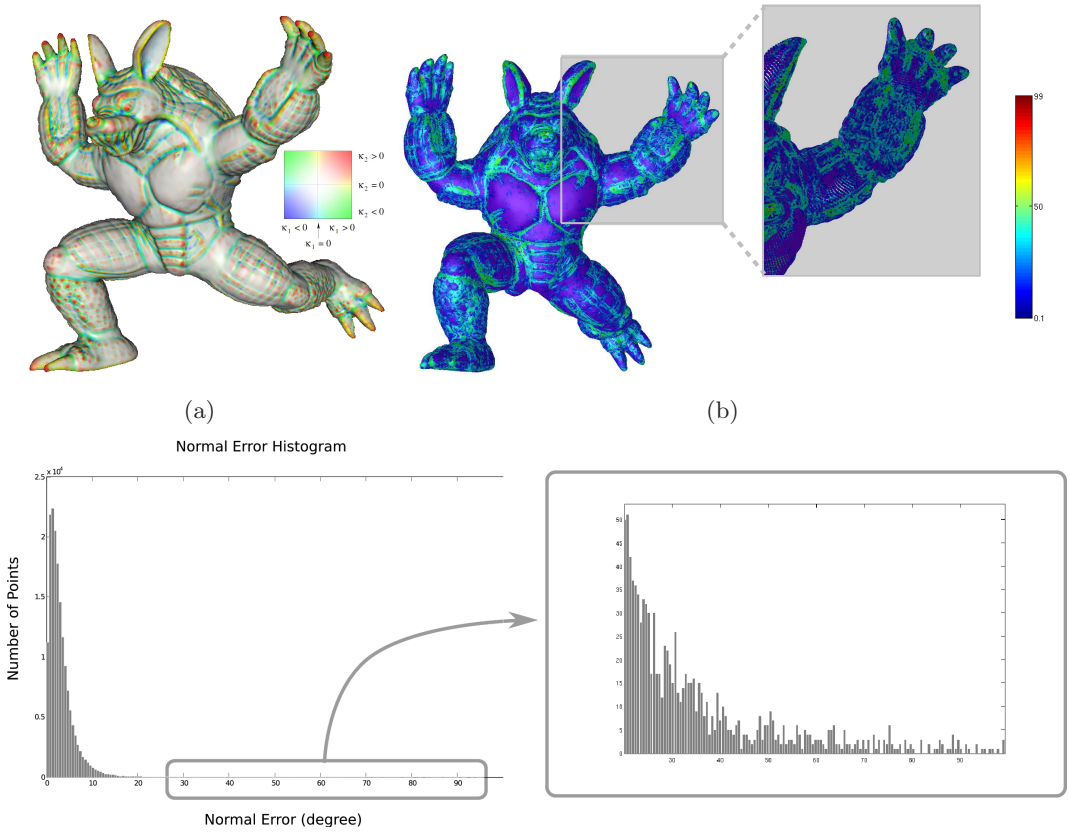



**Figure 2.9:** *Armadillo normal evaluation. We used the k-nearest neighborhood system with $k = 7$. (a) Principal curvatures colormap [Rus04]. The principal directions colormap is quite smooth, which indicates low noise level on the dataset. (b) Normal error colormap, (c) Normal error histogram, where the error is the angle between the reference and estimated normal vectors, expressed in degrees.*

### Smoothing effect

The first experiments aim at evaluating the performance of the normal estimation algorithm when the model is represented by a large dataset, with non-uniform points distribution, noisy, high points density, and composed by piece-wise smooth surfaces.

Figure 2.9 shows the result when applying the normal estimation algorithm to a modified version of the non-manifold Stanford Armadillo [Repb]. For the complete Armadillo model, composed of 165.954 points and low noise level, the mean normal error is $2,47$ degrees, the standard deviation is 3.71 degrees, and we have $9,65\%$ of these points that have their normal error estimation larger than 5 degrees. The first observation is that, for flat, low curvature, and low noise level regions, the estimated and the reference normals perform comparably, which verifies the assumption made about the neighborhood being homeomorphic to a disk. However, when such hypothesis are not verified, the estimate is less accurate, which correspond to regions of larger curvature.

In Figure 2.10 we evaluate the performance of the normal estimation algorithm when the dataset is corrupted with noise. We take the complete model of the INRIA Chinese lion [Repa], where, besides the large noise level, it is also observed a high surface roughness (fig. 2.11(a)).

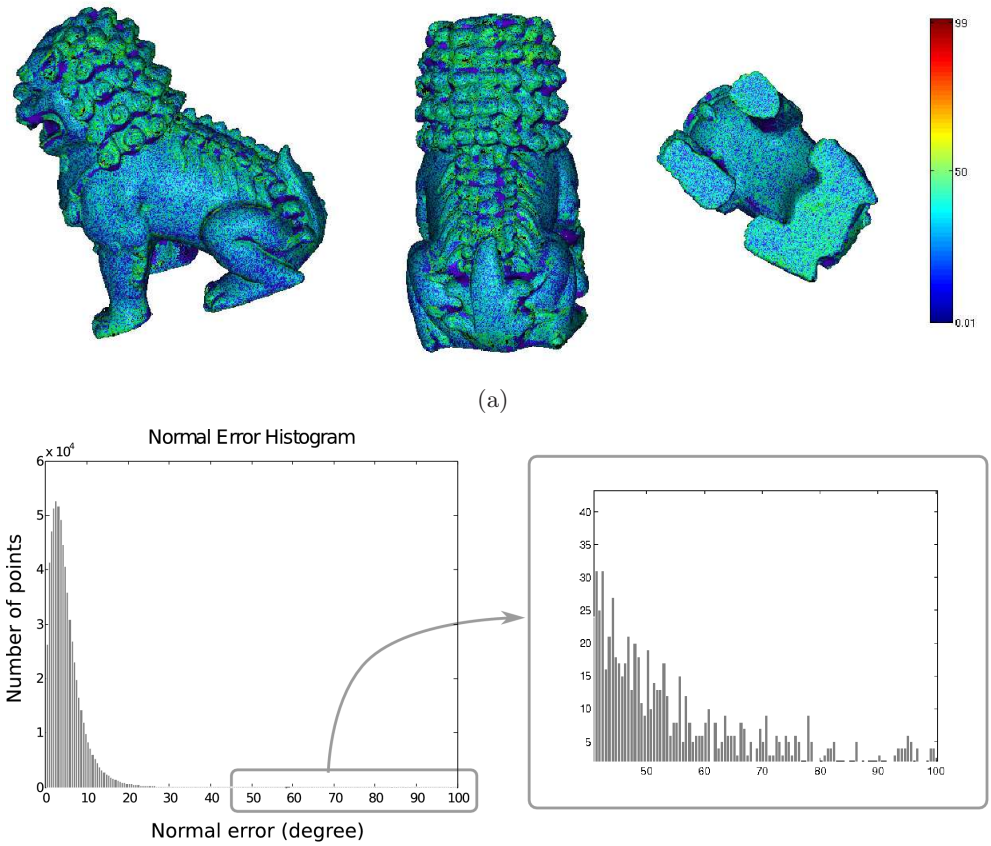
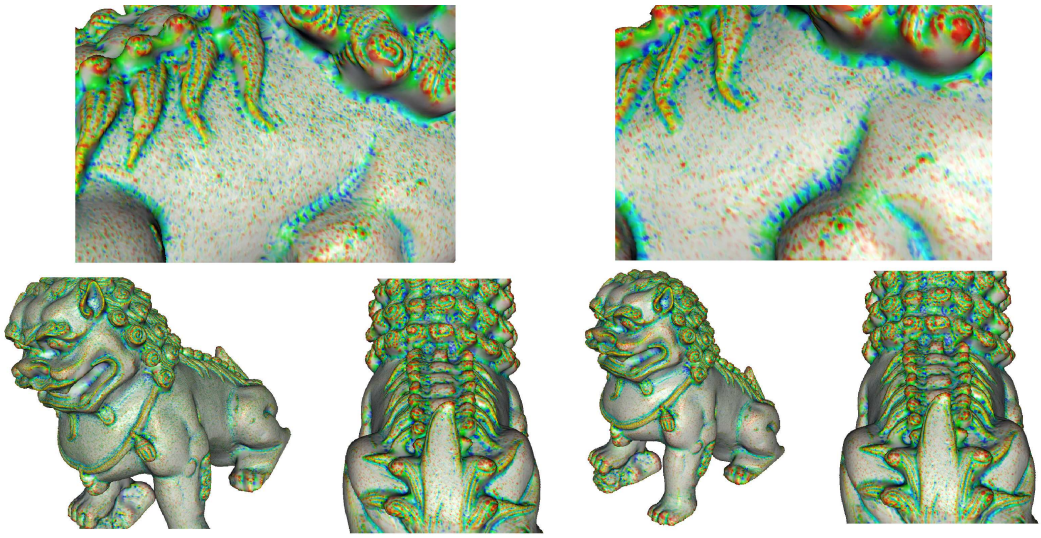



**Figure 2.10:** *Chinese dragon normal vector evaluation. (a) Normal error colormap, and (b) normal error histogram, where the error is the angle between the reference and estimated normal vectors, expressed in degrees.*

For such type of dataset, it is clear the normals smoothing effect when they are estimated using Hoppe's method. Again, we used the $k$-nearest neighborhood system, with $k = 7$, and it results in 9.47% of the dataset having normal error above 10 degrees. The mean and the standard deviation error of the estimated normals are 5.01 and 4.48 degrees, respectively.

At this point, we want to point out the difficulty of estimating accurately the normals and being robust to noise. The model in Figure 2.10 illustrates well such challenge. Note that surface roughness, noise, and sharp features mingle, and their differentiation on model fitting algorithms is nearly impossible. The observed normal error is not only due to inaccuracy of the estimation on sharp features, but also to the low-pass filtering effect on rough and noisy surfaces.

The smoothing of normals on sharp features, noise and roughness regions is propagated to local shape descriptors estimation. Figure 2.11 compares the *principal directions* colormap when the *principal directions* are computed using the normals taken from the triangle mesh and estimated by Hoppe's approach. It can be seen that Hoppe's method smoothes both rough, noisy and high curvature regions (Figure2.11(b)). On rough and noisy regions (top detail in Figure 2.11(b)) normal estimate smoothing can be seen as data filtering and denoising. On high curvature regions (colored regions in Figure 2.11(b)), normal smoothing is propagated to *principal*



(a) (b)

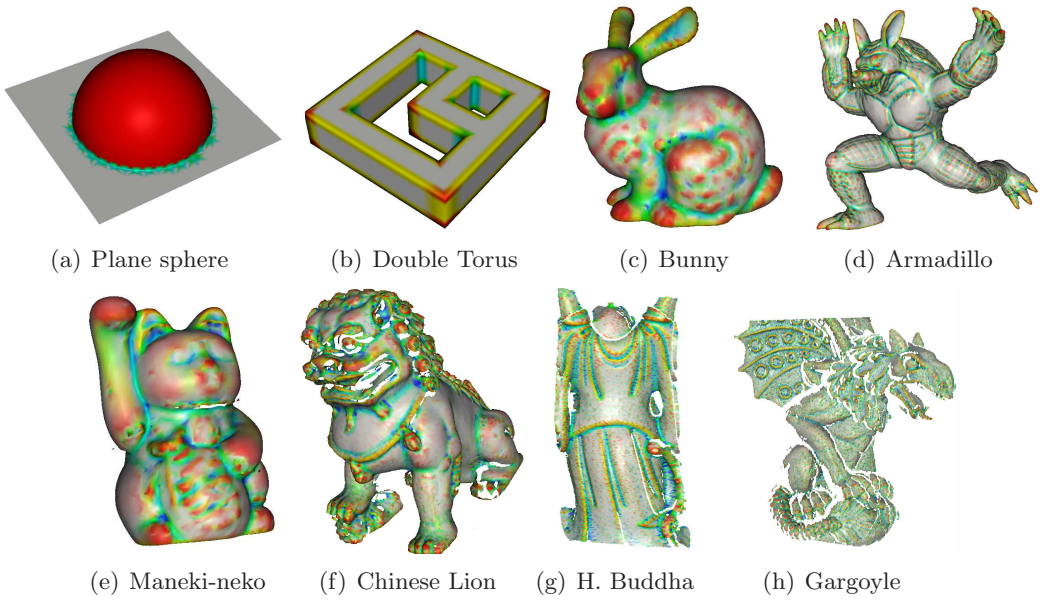
**Figure 2.11:** *Effect of the normal estimation smoothing on shape descriptor estimation. (a) Principal curvatures colormap when the normals are computed from the triangle mesh representation. (b) Principal curvatures colormap when normals are computed from the point cloud using Hoppe's approach.*

*directions* estimation. However, the smoothing effect did not impair data representation, once regions of high curvature preserved their high frequency features.

### The effect of the neighborhood size in the estimate

The normal accuracy is pronouncedly dependent on the neighborhood system and size. Other than the noisy point clouds, in this section, the normal estimation is performed on a set of various types of point clouds. The point clouds are illustrated in Figure 2.12. The main goal of this experiment is to verify the existence of a parameter interval in which the normal estimation error is small for a great variety of sampled surfaces. We will analyze the normal estimation algorithm performance in terms of the mean and the standard deviation errors. The point cloud can be categorized into synthetic, raw, and complete model datasets. Table 2.1 shows the specification of each dataset.

In Figures 2.13 we plot the mean and the standard deviation errors for different neighborhood sizes. The first and the second rows plot the results when the $k$-nearest and the ball neighborhood system are adopted, respectively. For small neighborhood size, we observe a large standard deviation and the mean error changes considerably for a small neighborhood size change. For such small neighborhoods the normal estimation tends to be unstable, independently of the neighborhood system used. For too big neighborhood, over smoothing degenerate the normal estimation of high frequency features. As expected, there is a neighborhood size interval where the average and the standard deviation estimation error stabilize and it provides normals that fairly fit the local surface. This result also verifies the stability of input parameter to the image choice. It means that if it is chosen a neighborhood size inside this interval, the estimation approximates the actual normal for a variety of point clouds, with different sampling densities and noise levels.



(a) Plane sphere (b) Double Torus (c) Bunny (d) Armadillo

(e) Maneki-neko (f) Chinese Lion (g) H. Buddha (h) Gargoyle

**Figure 2.12:** *Models used to evaluate the neighborhood influence on normal estimation. Principal curvatures colormap [lib].*

### Robustness to noise evaluation

In the final experiment, we test the robustness of normal estimation algorithm when the data is corrupted with noise. We add a Gaussian noise to the original sampled surface, with zero mean and where the standard deviation is taken to be a factor of the largest side of an axis parallel bounding box of the point cloud. This global scale for perturbations is perhaps more close to reality [DLS05].

In Figures 2.14 and 2.15 we plot the average errors and the standard deviation over two point clouds with different noise levels, for different neighborhood sizes. We show the results for the Happy Huddha and the Gorgoyle datasets. We observe that in presence of low noise level the algorithm performs well and the average error is close to noiseless performance. The neighborhood size in this scenario plays an important role in filtering noise and the proper choice of this parameter improves the performance of the algorithm. However, independently of the neighborhood size, fixed neighborhood is not robust when the dataset is heavily corrupted by a severe noise level.

A last observation concernes the robustness of the estimate when the dataset is represented by a non-uniform sampling. As the density of the sampling gets higher, the 2-manifold hypothesis hold true almost everywhere, even in regions where the data is sparser. Consequently, the normal estimation is accurate everywhere. Such behaviour is particularly observed in the Happy Boudha dataset (Figure 2.14), where the normals where accuratly estimated everywhere, even on the *scans* boundary.

| Image name | Type of data | Number of points | Dense | Uniform sampling | Manifold | Noisy |
|---|---|---|---|---|---|---|
| Plane Sphere | Parametric surface | 2515 | No | No | No | No |
| Double Torus | Parametric surface | 2686 | No | No | No | No |
| Bunny | Complete model | 35947 | Yes | Yes | Yes | No |
| Armadillo | Complete model | 165954 | Yes | Yes | No | No |
| Maneki-neko | Complete model | 9345 | No | Yes | Yes | No |
| Chinese Lion | point cloud | 67178 | Yes | No | No | Yes |
| Happy Buddha | point cloud | 63535 | Yes | No | Yes | Yes |
| Gargoyle | poinr clous | 86286 | Yes | No | No | Yes |

**Table 2.1:** *Specifications of the datasets used to evaluate the normal estimation algorithm.*

### 2.5.5 Summary of normal estimation algorithm performance

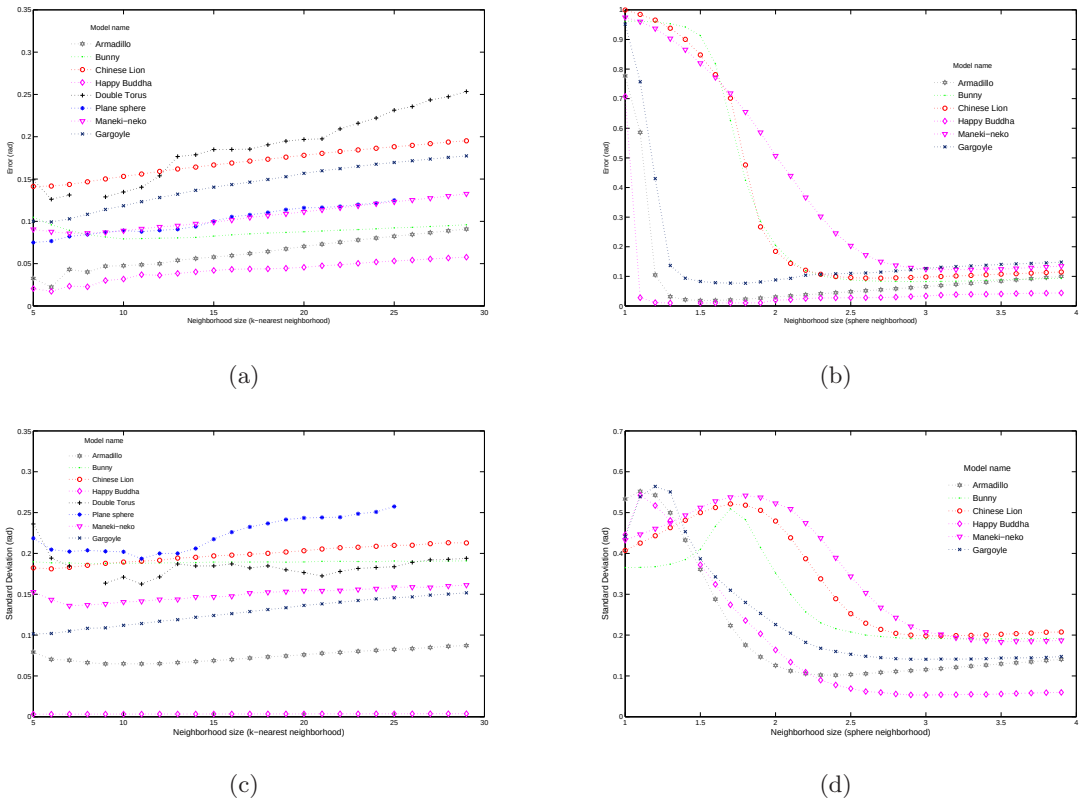
Comparing the results for different neighborhood systems and noise levels, we note that $k$-nearest neighborhood is preferable, once it produces more stable estimates for small neighborhoods. This is because $k$-nearest adapts the neighborhood size to the local point density, while the ball neighborhood implicitly assumes uniform sampling.

The use of a fixed neighborhood size provides an accurate estimation when the point cloud is dense, corrupted with measurement noise, or low noise level, non-uniform sampling and composed by low curvature surfaces. When such conditions are not satisfied, the estimation degenerates, once the neighborhood does not adapt to feature size. Of course, this problem can be attributed to the poor sampling density at the high curvature regions. The same is observed when an adaptive neighborhood approach is used, and both normal estimation methods may remove or smooth sharp features as well. In such scenario, and when the size of the point cloud is large and with high-resolution, the adaptive neighborhood approach may be expensive and may provide little improvement over the fixed neighborhood.

## 2.6 Curvature estimation

Curvature is the most used geometric intrinsic property to characterize a sampled surface. Assuming the point set $X$ is sampled from some surface $\Sigma$, the curvature of a surface at the point $\mathbf{x}_i$ in some direction $\mathbf{t}$ is a number which describes the deviation of the curve on a normal slice of the surface in $\mathbf{t}$ in the very close neighborhood of this point. The curvature can be seen as the reciprocal of the radius of the circle that best approximates a normal slice of the surface in that direction.

There is a large number of local shape descriptors that embed curvature information ([FJ89, Tau95, SH02, GMGP05, UH08]). Similar to normal estimation evaluation, a curvature estima-




(a) (b)

(c) (d)

**Figure 2.13:** *Normal estimation errors (in radians) under different neighborhood sizes, using (a),(b) k-nearest, and (c),(d) ball neighborhood systems.*

tion algorithm should be robust to noise and density variation in order to be applicable directly to raw point clouds. To this end, in this section we will extend two curvature estimation methods to work on raw point cloud: the polyhedral approximation of Taubin in [Tau95] and numerical approximation of Flynn *et al.* in [FJ89]. They were originally designed to work on triangle meshes [Tau95] and range images [FJ89], and they provide a complete and compact shape representation. Before we present these algorithms, we will present the basic concepts about the curvature estimation, and some of the related works.

### 2.6.1 Basic concepts

Let $\alpha : [a, b] \to \mathbb{R}^3$ be a curve parametrized by arc lenght $s$. The differentiable map $\alpha$ associates each $s \in (a, b)$ into a point $\alpha(s) = \mathbf{x}(s) \in \mathbb{R}^3$. Since the tangent vector, $\alpha'(s)$, has unit length, the norm $\|\alpha''(s)\|$ of the second derivative measures the rate of change of the angle which neighboring tangents make with the tangent at $s$. The normal norm $\|\alpha''(s)\|$ gives, therefore, a measure of how rapidly the curve pulls away from the tangent line at $s$. The curvature is defined as follows [dC76]:

**Definition 2.28** *Let $\alpha : [a, b] \to \mathbb{R}^3$ be a curve parametrized by arc length $s \in [a, b]$. The number $\|\alpha''(s)\| = \kappa$ is called the curvature of $\alpha$ at $s$.*

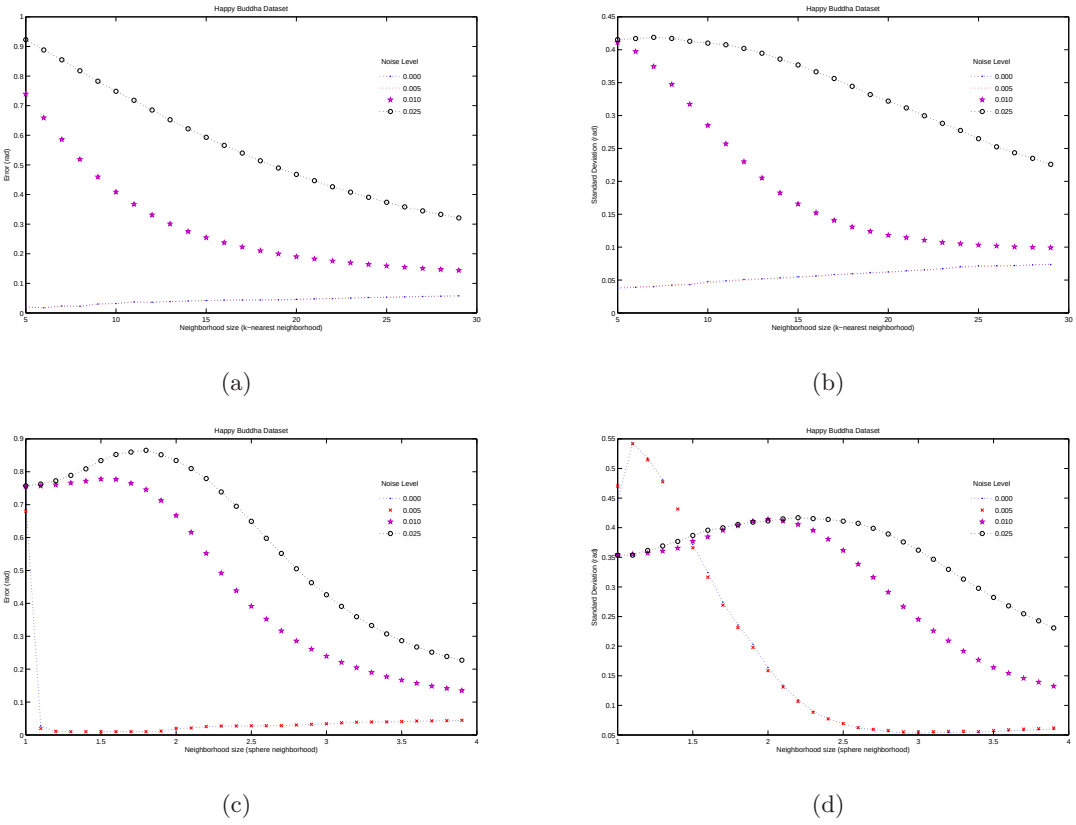



**Figure 2.14:** *Normal estimation errors and standard deviation (in radians) under noise for the Happy Buddha model, using (a),(b) k-nearest, and (c),(d) ball neighborhood systems.*

At a point $\kappa(s) \neq 0$, the normal vector $\mathbf{n}(s)$ in the direction of $\alpha''(s)$ is well defined by the equation $\alpha''(s) = \kappa(s)\mathbf{n}(s)$.

Given a surface $\Sigma$, the curvature $\kappa$ is defined for any parametric curve $\alpha$ defined on the $\Sigma$. Let $\mathbf{t}$ be a unit length tangent vector of the surface $\Sigma$ at a point $\mathbf{x}$. The *directional curvature* $\kappa_{\mathbf{x}}(\mathbf{t})$ is the curvature of the curve projected onto the plane containing the vector $\mathbf{t}$ and the surface normal $\mathbf{n}$. The *directional curvature* $\kappa_{\mathbf{x}}(\mathbf{t})$ of a surface $\Sigma$ at a point $\mathbf{x}$ in the direction of a unit length tangent vector $\mathbf{t}$ is defined by $\alpha''(0) = \kappa_{\mathbf{x}}(\mathbf{t})\mathbf{n}$, where $\alpha(s)$ is the normal section to $\Sigma$ at $\mathbf{x}$ parametrized by arc length, and such that $\alpha(0) = \mathbf{x}$ and $\alpha'(0) = \mathbf{t}$.

The *tensor curvature* of the surface $\Sigma$ is the map that assigns each sample point $\mathbf{x}$ to the function that measures the *directional curvature* $\kappa_{\mathbf{x}}(\mathbf{t})$. The *directional curvature* function is a quadratic form, i.e, it satisfies the identity [Tau95]:

$$\kappa_{\mathbf{x}}(\mathbf{t}) = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}^t \begin{pmatrix} \kappa_{\mathbf{x}}^{11} & \kappa_{\mathbf{x}}^{12} \\ \kappa_{\mathbf{x}}^{21} & \kappa_{\mathbf{x}}^{22} \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} t_1 & t_2 \end{pmatrix} \Pi \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \tag{2.29}$$

where $\mathbf{t} = t_1\mathbf{t}_1 + t_2\mathbf{t}_2$ is a tangent vector to $\Sigma$ at $\mathbf{x}$, $\{\mathbf{t}_1, \mathbf{t}_2\}$ is an orthonormal basis of the tangent space to $\Sigma$ at $\mathbf{x}$, $\kappa_{\mathbf{x}}^{11} = \kappa_{\mathbf{x}}(\mathbf{t}_1)$, $\kappa_{\mathbf{x}}^{22} = \kappa_{\mathbf{x}}(\mathbf{t}_2)$, and $\kappa_{\mathbf{x}}^{12} = \kappa_{\mathbf{x}}^{21}$. The symmetric matrix $\Pi$ appearing here, is known as the Weingarten matrix or the second fundamental tensor. Figure 2.16 illustrate the *principal curvatures* on a surface $\Sigma$.

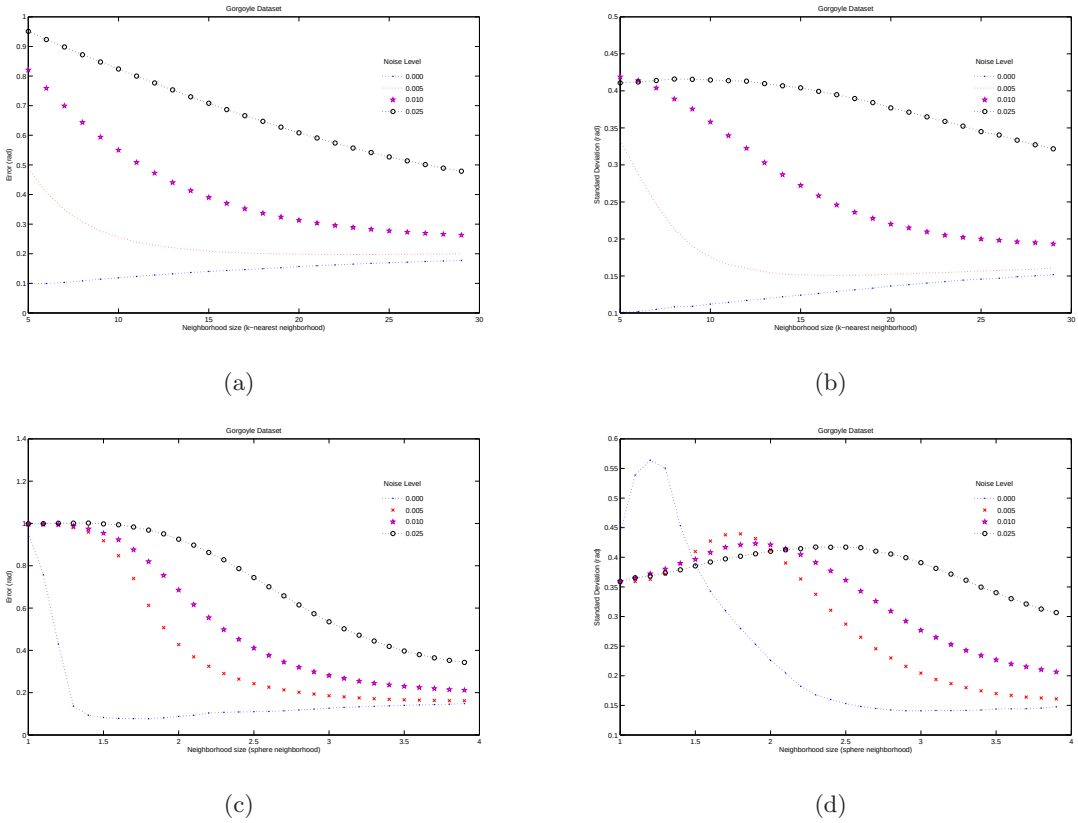



**Figure 2.15:** *Normal estimation errors and standard deviation (in radians) under noise for the Gorgoyle model, using (a),(b) k-nearest, and (c),(d) ball neighborhood systems.*

From all possible tangent vectors, there are two vectors in the tangent plane in which the *directional curvatures* are the largest and the smallest. This configuration occurs when $\kappa_{\mathbf{x}}^{12} = \kappa_{\mathbf{x}}^{21} = 0$ . Hence, $\kappa_{\mathbf{x}}^{11}$ and $\kappa_{\mathbf{x}}^{22}$ are the principal curvatures, which we will name $\kappa_{\mathbf{x}}^{1}$ and $\kappa_{\mathbf{x}}^{2}$ for simplicity. The corresponding $\{\mathbf{t}_1, \mathbf{t}_2\}$ are called the *principal directions* of $\Sigma$ at $\mathbf{x}$.

Two other curvature quantities, the mean curvature and the Gaussian curvature, can be derived from the principal curvatures. Together, the mean and Gaussian curvatures provide a complete and compact local shape representation. The mean curvature is defined by

$$H = (\kappa_{\mathbf{x}}^{1} + \kappa_{\mathbf{x}}^{2})/2 \tag{2.30}$$

and its value is closely related to the first variation of the surface area, which depends on the embedding. The Gaussian curvature is defined as

$$K = \kappa_{\mathbf{x}}^{1} \cdot \kappa_{\mathbf{x}}^{2} \tag{2.31}$$

and its value informs whether a surface is locally convex (when it is positive) or locally saddle (when is negative).

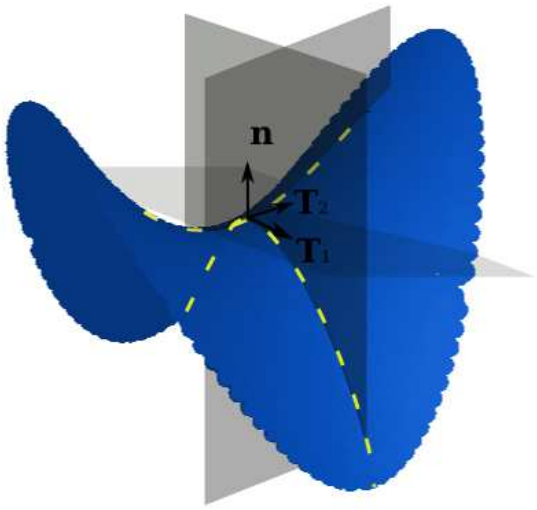



**Figure 2.16:** *Surface with tangent vector to* $\Sigma$ *at* $\mathbf{x}$, $\mathbf{t}_1$, $\mathbf{t}_2$, *in direction of the principal directions.*

### 2.6.2 Related works on *principal curvatures* estimation

To our knowledge, there are no *principal curvatures* estimation algorithms proposed in the literature designed to operate directly on the raw point clouds. However, the problem of extracting curvature information from an oriented point cloud has been extensively studied in computer vision and graphics [AM98, CP03, CGR$^+$04, GI04, HT03, MYHS04, AR05, TT05, YLHP06, YL89, CW00]. The existing methods for estimating *principal curvatures* and *directions* can be classified into four general categories:

**The Normal Curvature Approximation Methods:** Given a triangle mesh surface approximation, algorithms falling in this category computes the *directional curvature* for each edge leaving a point and use this estimate to find the second fundamental tensor. Given an approximation of the second fundamental tensor, the *principal curvatures* can be computed from the diagonalization of this matrix [Tau95, GI04], or fitting it using least squares [CS92, SH02], or a variation of these algorithms [KSPA01].

**Model fitting Methods:** Algorithms falling in this category, fit a small neighborhood around a point into some analytic surface model, and then compute the *principal curvatures* from the surface model [CP03, FJ89]. When the local shape at a point is from the same class of the surface being fit, these methods produce the exact result. However, on high curvature regions and near degenerate configurations, most notably if the points lie near a pair of intersecting lines, these algorithms in this category becomes unstable. Flynn in [FJ89] presents some techniques to estimate the *principal curvatures*, from surface fitting to numerical approaches. Goldfeather and Interrante [GI04] have shown that the instability of model fitting methods can be avoided by adding normals in the fit. Goldfeather and Interrante [GI04] have also showed that the quadratic surface method is identical to the normal curvature method, except that curvature

approximations are done using parabolas rather than circles.

**Tensor averaging methods:** The algorithms in this category compute the average of the curvature tensor over a small area of the polygonal mesh [Rus04], to find the second fundamental tensor at a point. The contribution of the adjacent faces at a point is averaged and the *principal curvatures* are taken as the eigenvalues of the approximate second fundamental tensor. The algorithms in this category cannot be applied to point clouds, because they rely on properties of the triangle faces.

**Integral invariants:** The main drawback of computing curvature based on numerical differentiation is its sensitivity to noise. Pottmann in [PWY+07, PWHY09] proposed an integral invariant shape descriptor, which is equivalent to curvature, to solve this problem. It computes the volume of the intersection of a small ball with the interior of the object defined by the input point set, which is proportional to the mean and Gaussian curvatures. The estimation of integral invariant shapes is supposed to be more robust due to its integral, instead of differential approach. However, most of application still have a denoising and smoothing prior to numeric computation, to guarantee repeatability of the estimate [AMCO08].

Any other curvature estimation method that does not rely on the properties of triangle mesh representation, such as face area, can be extended to work on unstructured point clouds. This is done by replacing the triangle mesh surface representation to a *neighborhood graph* representation. From the methods listed above, we have chosen two standard methods for estimating the curvature at a point: Taubin's normal curvature approximation method [Tau95] and Flynn's numerical approximation method [FJ89]. They were chosen because of their well established theoretical basis, and their efficiency to be computed and compared. In the next section we will detail each approach, and explicit how we adapt them to work on unstructured point clouds.

### 2.6.3 Curvature estimation using a normal curvature approximation approach

Taubin in [Tau95] proposed an algorithm to estimate the *principal curvatures* and *principal directions* of triangle mesh surface representation. The algorithm expresses the second fundamental tensor (2.29) at each point as an integral, constructed in time proportional to the neighborhood size. The integral $M_{\mathbf{x}}$ is a matrix equivalent to the three dimension *directional curvature* tensor (2.29) to non-tangent directions. It has the same eigenvectors of $\Pi$, and their eigenvalues are related by a fixed homogeneous linear transformation. The complete Taubin's method is presented in Appendix A.

Given a point $\mathbf{x}_i$ and its neighbors $\mathcal{N}_{r_i}(\mathbf{x}_i)$, the matrix $\mathbf{M}_{\mathbf{x}}$ can be approximated by a weighted sum over the neighborhood $\mathcal{N}_{r_i}(\mathbf{x}_i)$ as:

$$\hat{\mathbf{M}}_{\mathbf{x}_i} = \sum_{\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)} w_{ij} \kappa_{ij} \mathbf{t}_{ij} \mathbf{t}_{ij}^t \tag{2.32}$$

where $\kappa_{ij}$ is the *directional curvature* when the tangent vector $\mathbf{t}_{ij}$ is the unit length projection of

the vector $\mathbf{x}_j - \mathbf{x}_i$ onto the tangent plane. The tangent plane is defined as the plane perpendicular to the normal vector $\mathbf{n}_{\mathbf{x}_i}$. The matrix $\hat{\mathbf{M}}_{\mathbf{x}_i}$ is averaged by the weight $w_{ij}$, which, originally in [Tau95], is proportional to the surface area of all triangles incident to both $\mathbf{x}_j$ and $\mathbf{x}_i$. To compute the *principal curvatures* from (2.32) we need to first restrict the $3 \times 3$ matrix to the tangent plane to $\Sigma$ at $\mathbf{x}_i$. Finally, the eigenvalues and the eigenvectors of the resulting $2 \times 2$ matrix give the *principal curvatures* and *principal directions* of (2.29).

Note that, to compute the matrix $\hat{\mathbf{M}}_{\mathbf{x}_i}$ (2.32), we need to estimate, additionally, the tangent vector $\mathbf{t}_{ij}$ and the *directional curvature* $\kappa_{i,j}$ for each pair $(\mathbf{x}_i, \mathbf{x}_j)$. First, the tangent vector $\mathbf{t}_{ij}$ can be computed simply as the projection of the vector $(\mathbf{x}_j - \mathbf{x}_i)$ on the tangent plane $\langle \mathbf{n}_{\mathbf{x}_i} \rangle^{\perp}$:

$$\hat{\mathbf{t}}_{ij} = (\mathbf{x}_i - \mathbf{x}_j) + \beta \cdot \hat{\mathbf{n}}_{\mathbf{x}_i} \tag{2.33}$$

where the constant $\beta$ is given as $\beta = -\hat{\mathbf{n}}_{\mathbf{x}_i}^t(\mathbf{x}_i - \mathbf{x}_j)$ and the normal vector $\hat{\mathbf{n}}_{\mathbf{x}_i}$ is estimated using Hoppe's technique [HDD+92]. The tangent vector $\mathbf{t}_{ij}$ can then be estimated as:

$$\hat{\mathbf{t}}_{ij} = \frac{(\mathbf{I} - \hat{\mathbf{n}}_{\mathbf{x}_i}\hat{\mathbf{n}}_{\mathbf{x}_i}^t)(\mathbf{x}_i - \mathbf{x}_j)}{\|(\mathbf{I} - \hat{\mathbf{n}}_{\mathbf{x}_i}\hat{\mathbf{n}}_{\mathbf{x}_i}^t)(\mathbf{x}_i - \mathbf{x}_j)\|} \tag{2.34}$$

The second variable that must be computed is the *directional curvature*, $\kappa_{ij}$. The *directional curvature* is estimated from a parametric arc-length curve approximation. Given a *normal section* defined by the tangent vector $\mathbf{t}_{\mathbf{x}_i}$ and the normal $\mathbf{n}_{\mathbf{x}}$, the curve $\alpha$ parametrized by arc length $s$ (definition 2.28), such that:

$$\alpha(0) = \mathbf{x}_i \tag{2.35}$$

$$\alpha'(0) = \mathbf{t}_{\mathbf{x}_i} \tag{2.36}$$

$$\alpha''(0) = \kappa_{\mathbf{x}_i}(\mathbf{t}_{\mathbf{x}_i})\mathbf{n}_{\mathbf{x}_i} \tag{2.37}$$

It follows that the *directional curvature* can be computed by the discrete approximation for the arc-length in Laurent series up to the second order, we obtain:

$$\kappa_{\mathbf{x}_i}(\mathbf{t}_{\mathbf{x}_i}) = \lim_{s \to 0} \frac{2\mathbf{n}_{\mathbf{x}_i}^t(\alpha(s) - \mathbf{x}_i)}{\|\alpha(s) - \mathbf{x}_i\|^2} \tag{2.38}$$

If $\mathbf{x}_j$ is close from $\mathbf{x}_i$, however different from it, we can approximate the *directional curvature* as

$$\hat{\kappa}_{ij} = \frac{2\hat{\mathbf{n}}_{\mathbf{x}_i}^t(\mathbf{x}_j - \mathbf{x}_i)}{\|\mathbf{x}_j - \mathbf{x}_i\|^2} \tag{2.39}$$

We try to minimize noise effect by assigning a confidence measure to each *directional curvature* estimation. The only constraint is that the sum of the weight over the all neighbors must be equal to one.

The question of weighting on mesh representation is viewed as how much of the face curvature should be accumulated at each vertex, and prior works have been addressed to this problem [Rus04]. The weighting problem on point clouds has been addressed in contexts other than

**Algorithm:** Estimation of *principal curvatures* using Taubin's approach

**Input**: Unorganized point cloud $X = \{\mathbf{x}_i\} \in \mathbb{R}^3$ with $i = 1, \cdots, n$.

1. Construct the *neighborhood graph* $G_{\mathcal{N}}$, for a given sphere radius $r$ (defined through the sphere or $k$-nearest neighborhood system). This graph embeds points spatial connectivity.
2. Estimate the oriented normal vector for each point $\mathbf{x}_i \in X$ using Hoppe's technique [HDD+92] and Johnson orientation propagation algorithm [Joh97]. Note that it is used a different neighborhood size, radius $r$, to estimate such normals.
3. **for** $\mathbf{x}_i \in X$ **do**
4. Initialize the matrix $\hat{\mathbf{M}}_{\mathbf{x}_i} = \mathbf{0}_{3\times 3}$.
5. **for** $\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)$ **do**
6. Given $\mathbf{x}_i$, $\mathbf{x}_j$ and $\hat{\mathbf{n}}_i$, estimate the curvature $\hat{\kappa}_{ij}$ (2.39).
7. Estimate the tangent vector $\hat{\mathbf{t}}_{ij}$ (2.34)
8. *Compute $w_{ij}$ using (2.40)*
9. $\hat{\mathbf{M}}_{\mathbf{x}_i} = \hat{\mathbf{M}}_{\mathbf{x}_i} + w_{ij}\hat{\kappa}_{ij}\hat{\mathbf{t}}_{ij}\hat{\mathbf{t}}_{ij}^t$ (2.32).
10. **end for**
11. Let $\mathbf{e}_1 = (1, 0, 0)^t$. $\mathbf{Q}_{\mathbf{x}_i} = \mathbf{I} - \mathbf{w}_{\mathbf{x}_i}\mathbf{w}_{\mathbf{x}_i}^t$, where $\mathbf{w}_{\mathbf{x}_i} = \dfrac{\mathbf{e}_1 \pm \hat{\mathbf{n}}_{\mathbf{x}_i}}{\|\mathbf{e}_1 \pm \hat{\mathbf{n}}_{\mathbf{x}_i}\|}$, with a minus sign if $\|\mathbf{e}_1 - \hat{\mathbf{n}}_{\mathbf{x}_i}\| > \|\mathbf{e}_1 + \hat{\mathbf{n}}_{\mathbf{x}_i}\|$ , and plus sign otherwise.
12. Compute $\mathbf{Q}_{\mathbf{x}_i}\hat{\mathbf{M}}_{\mathbf{x}_i}\mathbf{Q}_{\mathbf{x}_i}^t$ to restrict $\hat{\mathbf{M}}_{\mathbf{x}_i}$ to the tangent plane.
13. The *principal curvatures* $\kappa_{\mathbf{x}}^1$ and $\kappa_{\mathbf{x}}^2$ are computed from the relations in Appendix A.
14. **end for**

**Table 2.2:** *Estimation of principal curvatures using Taubin's approach.*

curvature estimation, such as discrete fairing [Pau03], reconstruction [AK04], among others. In [YKS00], they use the *principal curvatures* algorithm proposed by Taubin on point clouds and they set the weight equally $w_{ij} = \frac{1}{|\mathcal{N}_{r_i}(\mathbf{x}_i)|}$. A more adaptive weight should take the angle between normals into account because, in the *directional curvature* approximation, it is made an assumption that locally, the curvature is smooth. Hence, the normals cannot vary abruptly. We use the following weighting Gaussian function

$$w_{ij} = e^{-h_i \cdot (\angle(\mathbf{n}_{\mathbf{x}_i}, \mathbf{n}_{\mathbf{x}_j}))^2} \tag{2.40}$$

where $h_i$ is a scale factor and $(\angle(\mathbf{n}_{\mathbf{x}_i}, \mathbf{n}_{\mathbf{x}_j}))^2$ is the square of angle between the normals. This weighting is applied in (2.32) to compute the integral matrix. The scale factor $h_i$ is taken proportional to the cone angle, defined as

$$h_i = h \cdot \max_{\mathbf{x}_j \in \mathcal{N}_{r_i}} \left(\angle(\mathbf{n}_{\mathbf{x}_i}, \mathbf{n}_{\mathbf{x}_j})\right) \tag{2.41}$$

where $h$ is a user-defined constant. The complete *principal curvatures* estimation algorithm proposed by Taubin in [Tau95] and adapted here to work on point clouds is summarized in table 2.2.

### 2.6.4 Curvature estimation using a numerical approximation approach

Flynn *et al.* [FJ89] estimates the *principal curvatures* at a point $\mathbf{x}_i$ by considering the normal variation between $\mathbf{x}_i$ and its neighbors. The strategy adopted is to estimate an approximate of the *directional curvature* for each pair points $\mathbf{x}_i$ and $\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)$ and take the *principal curvatures* as the maximum and minimum *directional curvatures*. It is assumed that the normal vectors are oriented, consistent, and the origin of the reference frame is located at the point of interest and the tangent plane is the one perpendicular to the normal vector.

From curvature definition 2.28 the *directional curvature* at a point $\mathbf{x}$ in direction of $\alpha'(0) = \mathbf{t}$ can be approximated by:

$$\kappa_{\mathbf{x}}(\mathbf{t}) = \lim_{s \to 0} \frac{\|\alpha'(s) - \alpha'(0)\|}{\|\alpha(s) - \alpha(0)\|} \tag{2.42}$$

where $\alpha(0) = \mathbf{x}$.

Given two close points $\mathbf{x}_i$ and $\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)$, the *directional curvature* (2.42) can be computed as:

$$\hat{\kappa}_{ij} = \frac{\|\mathbf{t}_j - \mathbf{t}_i\|}{\|\mathbf{x}_j - \mathbf{x}_i\|} \tag{2.43}$$

where $\mathbf{t}_j$ and $\mathbf{t}_i$ are the tangent vectors at $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively, and they lie on the curve $\alpha(s)$. Since both tangent vectors and normals have unit length, and $\mathbf{t}_j$ is perpendicular to $\mathbf{n}_j$, such as $\mathbf{t}_i$ to $\mathbf{n}_i$, we have that:

$$\|\mathbf{t}_j - \mathbf{t}_i\| = \|\mathbf{n}_j - \mathbf{n}_i\| \tag{2.44}$$

The *directional curvature* of $\mathbf{x}_i$ in direction of a neighbor $\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)$ is then given as a discrete approximation to one-dimensional curvature along the hypothesized curve from $\mathbf{x}_i$ to $\mathbf{x}_j$ [FJ89]:

$$\hat{\kappa}_{ij} = \begin{cases} \dfrac{\|\mathbf{n}_i - \mathbf{n}_j\|}{\|\mathbf{x}_i - \mathbf{x}_j\|} & \text{if} \quad \|\mathbf{x}_i - \mathbf{x}_j\| \leq \|(\mathbf{n}_i + \mathbf{x}_i) - (\mathbf{n}_j + \mathbf{x}_j)\| \\ -\dfrac{\|\mathbf{n}_i - \mathbf{n}_j\|}{\|\mathbf{x}_i - \mathbf{x}_j\|} & \text{otherwise} \end{cases} \tag{2.45}$$

where the *directional curvature* is positive for locally convex and negative for locally saddle curves.

The algorithm proposed by Flynn does not estimate the actual *principal curvatures*. Instead, the *principal curvatures* are taken as the maximum and the minimum of (2.45) computed for all $\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)$. This is a numerical approximation which is accurate for noiseless, dense and uniformly distributed point clouds. The resulting algorithm is linear, both in time and in space, as a function of the number of vertices and neighborhood size. Since the method estimates the curvature from differential invariant properties of the surfaces, it amplifies the noise effect. Flynn's curvature estimation algorithm is summarized in table 2.3.

### 2.6.5 Similarities and differences between Taubin and Flynn's approaches

So far, we have presented two *principal curvatures* estimation algorithms, originally designed to mesh surface representation. Now, we show some properties of both *principal curvatures* estimation algorithms and how these algorithms can be modified in order to estimate the *principal curvatures* directly on point clouds.

#### *Directional curvature* equivalence

Both of the above algorithms incorporate different approximations for the *directional curvature* to estimate the *principal curvatures*. In order to compare them we need first to explicit the equivalence relationship between these two *directional curvatures* approximations (2.39) and (2.45). The *directional curvature* used by Taubin is defined as (2.39):

$$\hat{\kappa}_{ij} = \frac{2\hat{\mathbf{n}}_{\mathbf{x}_i}^t(\mathbf{x}_j - \mathbf{x}_i)}{\|(\mathbf{x}_j - \mathbf{x}_i)\|^2}$$

The *directional curvature* (2.39) is not a symmetric function, since $\hat{\kappa}_{ij} \neq \hat{\kappa}_{ji}$. Let $\mathbf{x}_i$ and $\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)$ be two points in the very close neighborhood. We define the *average directional curvature* as:

$$\overline{\kappa(\mathbf{x}_i, \mathbf{x}_j)} = \frac{\hat{\kappa}_{ij} + \hat{\kappa}_{ji}}{2} \tag{2.46}$$

By expanding the *average directional curvature* using Taubin approximation for the *direc-*

**Algorithm:** Estimation of *principal curvatures* using Flynn's curvature approximation

**Input**: Unorganized point cloud $X = \{\mathbf{x}_i\} \in \mathbb{R}^3$ with $i = 1, \cdots, n$.

1. Construct the *neighborhood graph* $G_{\mathcal{N}}$, for a given sphere radius $r$ (defined through the sphere or $k$-nearest neighborhood system). This graph embeds spatial connectivity of points.
2. Estimate the oriented normal vector for each point $\mathbf{x}_i \in X$ using Hoppe's technique [HDD+92] and Johnson orientation propagation algorithm [Joh97]. Note that it is used a different neighborhood size, radius $r$, to estimate such normals.
3. Initialize the *principal curvatures*, $\kappa^1_{\mathbf{x}_i} = -\infty$ and $\kappa^2_{\mathbf{x}_i} = \infty$.
4. **for** $\mathbf{x}_i \in X$ **do**
5. **for** $\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)$ **do**
6. **If** $\angle(\mathbf{n}_{\mathbf{x}_i}, \mathbf{n}_{\mathbf{x}_j}) < h$ (2.41)
7. Given $(\mathbf{x}_i, \mathbf{x}_j)$ and $(\hat{\mathbf{n}}_i, \hat{\mathbf{n}}_j)$, compute $\kappa_{ij}$ (2.45).
8. **If** $(\kappa_{ij} > \kappa^1_{\mathbf{x}_i})$
9. **then**
10. $\kappa^1_{\mathbf{x}_i} = \kappa_{ij}$
11. **If** $(\kappa_{ij} < \kappa^2_{\mathbf{x}_i})$
12. **then**
13. $\kappa^2_{\mathbf{x}_i} = \kappa_{ij}$
14. **end if**
15. **end for**
16. Assign to the point $\mathbf{x}_i$ the *principal curvatures*, $\kappa^1_{\mathbf{x}_i}$ and $\kappa^2_{\mathbf{x}_i}$.
17. **end for**

**Table 2.3:** *Estimation of principal curvatures using Flynn's curvature approximation.*

*tional curvature* (2.39), we have

$$\overline{\kappa(\mathbf{x}_i, \mathbf{x}_j)} = \frac{1}{2}\left(\frac{2\hat{\mathbf{n}}^t_{\mathbf{x}_i}(\mathbf{x}_j - \mathbf{x}_i)}{\|(\mathbf{x}_j - \mathbf{x}_i)\|^2} + \frac{2\hat{\mathbf{n}}^t_{\mathbf{x}_j}(\mathbf{x}_i - \mathbf{x}_j)}{\|(\mathbf{x}_i - \mathbf{x}_j)\|^2}\right) \tag{2.47}$$

$$= \frac{\hat{\mathbf{n}}^t_{\mathbf{x}_i}(\mathbf{x}_j - \mathbf{x}_i) - \hat{\mathbf{n}}^t_{\mathbf{x}_j}(\mathbf{x}_j - \mathbf{x}_i)}{\|(\mathbf{x}_j - \mathbf{x}_i)\|^2} \tag{2.48}$$

$$= \frac{(\hat{\mathbf{n}}^t_{\mathbf{x}_i} - \hat{\mathbf{n}}^t_{\mathbf{x}_j})(\mathbf{x}_j - \mathbf{x}_i)}{\|(\mathbf{x}_j - \mathbf{x}_i)\|^2} \tag{2.49}$$

$$= -\frac{(\hat{\mathbf{n}}_{\mathbf{x}_j} - \hat{\mathbf{n}}_{\mathbf{x}_i})^t(\mathbf{x}_j - \mathbf{x}_i)}{\|(\mathbf{x}_j - \mathbf{x}_i)\|^2} \tag{2.50}$$

In a very close neighborhood, where a small change on surface normal is observed, we have that

$$(\mathbf{n}_{\mathbf{x}_j} - \mathbf{n}_{\mathbf{x}_i}) \approx (\mathbf{x}_j - \mathbf{x}_i) \tag{2.51}$$

and, consequently,

$$(\mathbf{n}_{\mathbf{x}_j} - \mathbf{n}_{\mathbf{x}_i})^t(\mathbf{x}_j - \mathbf{x}_i) \approx \|\mathbf{n}_{\mathbf{x}_j} - \mathbf{n}_{\mathbf{x}_i}\|^2 \tag{2.52}$$

This approximation is accurate for highly dense point clouds, where we can assume the curvature is bounded, near constant, and close to zero (plane). In such scenario, the *average directional curvature* can be approximated by

$$\frac{\hat{\kappa}_{ij} + \hat{\kappa}_{ji}}{2} = -\frac{\|\mathbf{n}_{\mathbf{x}_j} - \mathbf{n}_{\mathbf{x}_i}\|^2}{\|(\mathbf{x}_j - \mathbf{x}_i)\|^2} \tag{2.53}$$

which is similar to Flynn's *directional curvature* approximation (2.45).

Lets call, for now, $\hat{\kappa}^F_{ij}$ and $\hat{\kappa}^T_{ij}$ the *directional curvatures* approximation defined by Flynn (2.45) and Taubin (2.39), respectively. From (2.53), we have that the relationship between the magnitude of these two approximations is given by

$$\frac{\hat{\kappa}^T_{ij} + \hat{\kappa}^T_{ji}}{2} = (\hat{\kappa}^F_{ij})^2 \tag{2.54}$$

For highly dense point clouds, where (2.51) holds true, $\hat{\kappa}^T_{ij} = \hat{\kappa}^T_{ji}$, and, consequently, $\hat{\kappa}^T_{ij} = \hat{\kappa}^F_{ij}$. Otherwise, Flynn's approximation in an average of Taubin's.

Both approaches amplify noise effect, since they are computed from differential quantities, such as normal and points distance. However, at high curvature and noisy regions, Flynn's *directional curvature* approximation is more sensitive than Taubin's. Given a point $\mathbf{x}_i$ and $\mathbf{x}_j$ for example, Taubin's approximation takes into account only the normal at $\mathbf{x}_i$, while Flynn's approach uses both normals.

There is another very important difference between these two approaches, which is the strategy adopted to estimate the *principal curvatures* from *directional curvature* estimation. While Flynn's approach compute the *principal curvatures* as the maximum and the minimum of the estimated *directional curvatures*, Taubin estimates the second fundamental tensor (2.29) to compute the *principal curvatures*. It means that in Taubin's approach, contrary to Flynn's approach, the *principal directions* do not necessarily have the same direction of the vector from $\mathbf{x}_i$ to any neighbor. Besides, there is no filtering embedded on Flynn's approach and noise can degenerate the estimate. Taubin, on the other hand, average the contribution of each estimated *directional curvature* on the integral matrix computation (2.39), which can be used to filter noisy estimates. Next, we resume the main properties of each *principal curvature* algorithm.

***Principal curvatures* estimation properties**

- *Complexity:* They are linear, both in time and in space, as a number of points and neighbors. This property makes them both suitable to estimate the local shape of large datasets.
- *Robustness to noise:* They amplify noise effect, since they are computed from differen-

tial quantities, such as normal and points distance. Flynn's approach, is, however, more sensitive to noise.

- *Principal directions:* They use different strategies to estimate the *principal directions* and *principal curvatures* from the *directional curvature* estimation. In Flynn's approach both *principal directions* and *principal curvatures* are actually taken as some of the *directional curvature*. However, in Taubin's approach, the *principal directions* and *curvatures* are the eigenvectors and eigenvalues of the integral matrix (2.32).

- *Input parameters:* Similar to normals estimation, the neighborhood size determines the scale on which the curvature invariant is computed. However, instead of trying to find the optimum neighborhood size to estimate the curvature invariants many algorithms [GMGP05, HP05, UH08] propose to compute a scale-space representation of such invariant, where the geometric descriptor is considered, for each point, at several scales simultaneously. The estimates are then analyzed in order to filter outliers and obtain the scale that gives a more consistent representation of the local shape. Such strategy has crucial relevance for accurate performance of feature-based tasks [UH08]. Taubin's approach has another parameter, which is the weight $w_{ij}$ that average the contribution of each *directional curvature* on the integral matrix computation (2.32).

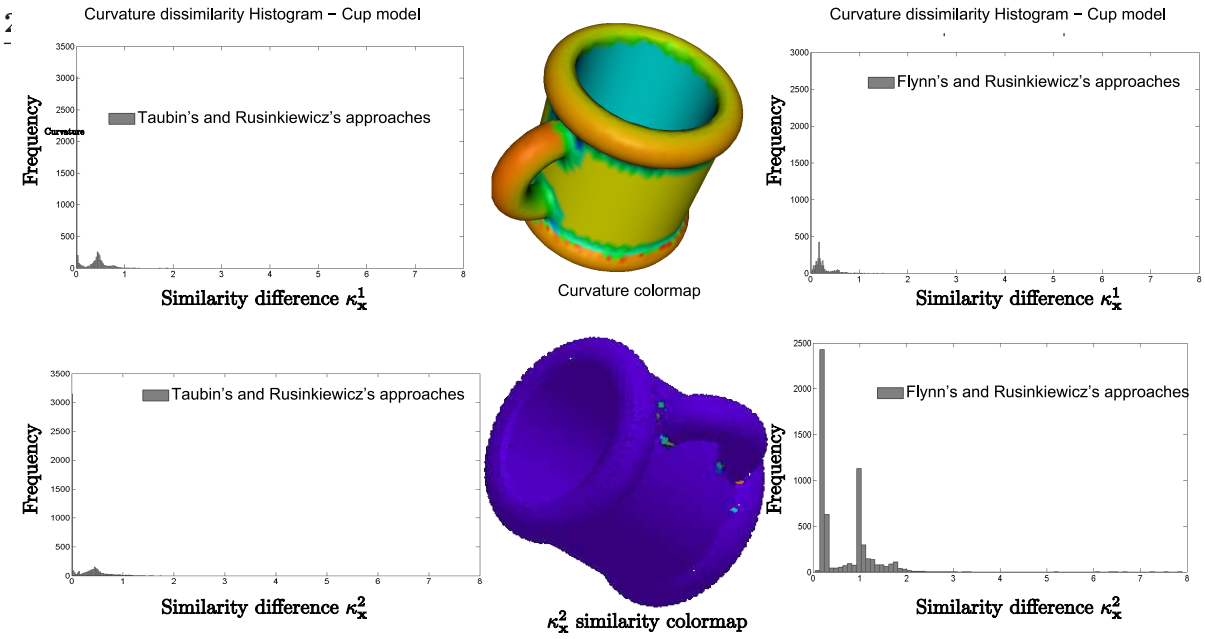
### 2.6.6 Curvature estimation evaluation

In this section we will evaluate Flynn's and Taubin's curvature estimation methods on unstructured point cloud. We differentiate between two categories of data: synthetic and real raw point clouds. While the interest for the synthetic data is generated from the fact that it is accurate and allows for ground truth to be produced at any point, the interest in raw point cloud is motivated by the fact that in most cases it is noisy, with direct influence on the accuracy and stability of the algorithms.

Algorithms 2.3 and 2.2 detail the procedure to estimate the *principal curvatures* from the point cloud in order to compute the *Gaussian* and *mean* curvature invariants. When evaluating the performance of *principal curvatures* algorithms, the normals are taken as an input data, in order to dissociate the problem of normal estimation to the problem of curvature estimation.

Rusinkiewicz in [Rus04] presents a tensor averaging method to estimate the *principal curvatures* and *principal directions* on irregular triangle meshes. This method relies on the well established properties of continuous surfaces represented by triangle meshes and it provides accurate estimates in the presence of irregular tessellation and moderate amounts of noise. Besides, it has no input parameter.

In Figure 2.17 we show the results that verify the equivalence of the estimates. It is compared the *principal curvatures* estimation approaches of Rusinkiewicz [Rus04], Flynn [FJ89] and Taubin [Tau95] in a synthetic dataset. The error histograms show the *principal curvatures* error when Rusinkiewicz's estimate is our reference. The cup model is an uniform sampling, noiseless, synthetic dataset. The cup *principal curvatures* colormap is showed in Figure 2.17(a). We take the curvature estimated using Rusinkiewicz's approach on the cup triangle mesh representation




(a)

**Figure 2.17:** *Principal curvatures comparison: $\kappa_{\mathbf{x}}^1$ , $\kappa_{\mathbf{x}}^2$ similarity, when they are estimated using Flynn's (right) and Taubin's (left) methods and compared with the estimation computed using Rusinkiewiewics's technique.*

as our reference. The normals computed from the mesh representation are used in all algorithms. The dissimilarity between the estimated $\kappa_{\mathbf{x}}^1$s and $\kappa_{\mathbf{x}}^2$s is plotted in Figure 2.17(b). The dissimilarity curvature histograms has zero error for most of the sample points, except for point on non-manifold regions.

In Figure 2.19 we show the comparison of the three methods over the Happy Buddha dataset. We want to evaluate the robustness to noise of the methods presented. The Happy Buddha is a raw point cloud with a nearly uniform sampling, and the $\kappa_{\mathbf{x}}^2$ dissimilarity colormap is illustrated in Figure 2.18. Note that all three methods produce good results for perfect data, but Taubin's and Flynn's approach degrade rapidly with the addition of noise. We have found that the quality of the curvature estimated using both Taubin and Flynn methods degenerates even when low noise level corrupts the dataset. This behavior is due to the amplification of noise, because of the use of differential quantities to estimate the curvature. This lack of robustness to noise compromises the quality of data characterization, and, consequently, the performance of the entire registration pipeline.

## 2.7 Conclusions

In this chapter, we presented the fundamental properties of raw point clouds, the assumptions made about noise, points distribution, and about the local surface properties. We presented Hoppe's normal estimation method, which gives a good approximation to the tangent plane around the point. We verified that the estimate is quite accurate for flat and low curvature surfaces, and robust to moderate noise level. We used a fixed neighborhood size to compute the normal and we evaluate the influence of the neighborhood system and size on the accuracy of the estimation. For a fixed neighborhood size, we observe that the Hoppe's method filters

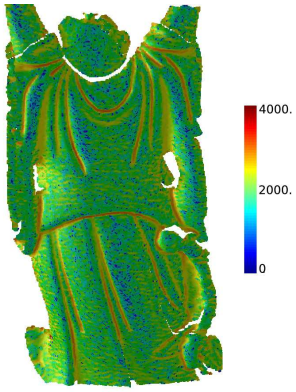



**Figure 2.18:** $\kappa^2_{\mathbf{x}}$ *dissimilarity colormap between Flynn's and Rusinkiewicz's curvature estimation methods.*

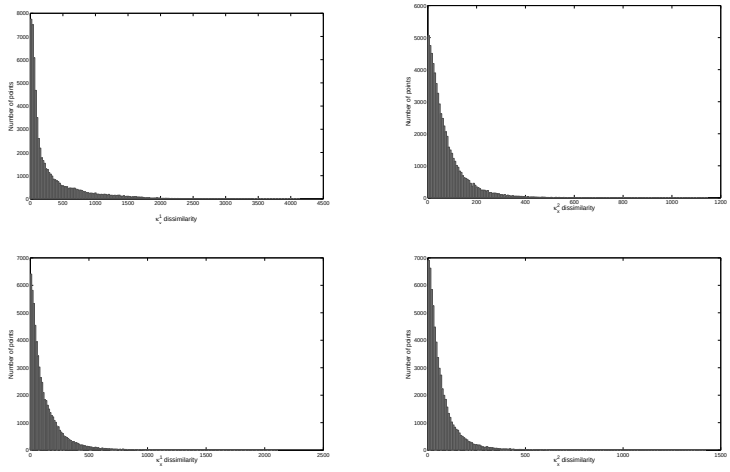


**Figure 2.19:** *Curvature dissimilarity histogram for* $\kappa^1_{\mathbf{x}}$ , $\kappa^2_{\mathbf{x}}$ *when they are estimated using Flynn's (a),(b) and Taubin's (left) methods (c)(d).*

noise but it also smooths high frequency feature, and, compared to the adaptive neighborhood approach proposed in [MN03], the smoothing is bigger. We believe that, since the difference between a fixed and an adaptive neighborhood size is small for low level noisy datasets, it does not justify to use an expensive method to have an comparably performance.

We also presented an extension of two methods to estimate the *principal curvatures* around a neighborhood to work directly on raw point clouds. Both methods were originally designed to work on triangle mesh surfaces, with a low noise level. We compared the *principal curvatures* estimation with Rusinkiewicz's method [Rus04]. We observed the estimations using both approaches [FJ89, Tau95] are based on differential quantities and an accurate curvature calculation in the presence of statistical outliers and noise is very difficult.

Using techniques presented so far, we can compute shape descriptors directly from raw point clouds and structure the point cloud according to neighborhood relationships. In the next chapter, we explore these estimates to reduce data volume and built a higher-level representation through segmentation.

# Chapter 3

# Segmentation of Unstructured Raw Point Clouds

## 3.1 Introduction

In this chapter we address the problem of representing a noisy raw 3D point cloud as a set of regions. Our approach falls within the framework of image segmentation, where one wants to partition the input data into a connected set of homogeneous regions. For this matter, we present a graph-based segmentation algorithm, a data-driven approach designed to partition directly large and raw 3D point clouds.

Segmentation is a broad term which refers to any method of grouping together points which are similar in a feature space, and there is a growing movement towards using segmentation to provide preliminary point grouping. When performing registration of large and raw point clouds, for example, low-level description, or local shape descriptor, is often an inadequate data representation for such task. This is because local shape descriptors matching does not guarantee one-to-one correspondence. Reflecting on the problem, segmentation can be then an intermediate phase, in which objective is mostly a substantial reduction in data volume without information loss. This compact data representation, given as a region set, can be used as the input feature set to a region-based registration scheme. Such strategy will be developed in Chapter 4.

We define our segmentation algorithm as a recursive bi-partition of the weighted *neighborhood graph*. Region homogeneity is inferred from the local feature analysis on a weighted graph constructed from the triplet: Euclidean neighborhood relationship, feature space defined by local shape descriptors and similarity function. We extended the 2D segmentation algorithm using the graph's Minimum Spanning Tree ($MST$), introduced in [Zah71b] and later improved in [FH04], to work directly on raw 3D point clouds. This graph-based method is a combinatorial

optimization algorithm, in which we extract the $MST$ of the *neighborhood graph* to decrease the search space of feasible segmentations. The $MST$ is a subgraph of the *neighborhood graph*, where the edges in the $MST$ are aligned with, or point towards, the closest isolevel in the feature space. We make very few restrictive assumptions about the shapes of the underlying sampled surface and the segmentation generates regions of general shapes using generic cues of coherence and affinity between points in feature space.

From the resulting segmentation, we derive a concise formulation for the region. In order to exploit the segmentation results in a larger system, we propose a set of requirements we consider fundamental and we also propose a taxonomy for 3D segmentation algorithms. Together, they guide to the choice of the most appropriate segmentation algorithm for our application. We use these concepts to validate our approach, to outline some properties of the region formed, and to compare our segmentation algorithm to other segmentation algorithms in the same category. Such an evaluation is used for both synthetic and raw point cloud segmentation.

## 3.2 Using 3D segmentation results in larger systems

(a) (b)

**Figure 3.1:** *Perceptual grouping problem: the figurines [ED] were segmented using two different strategies, but both producing acceptable results and they mimic different human perception. The models on left were segmented using our MST-based algorithm [ADC07b] and the models on right were segmented using a watershed segmentation algorithm [APFJ+08a].*

Adhering to the notation from the previous chapter, let $X$ be a set of points representing the sampled surface. We define $\mathcal{P}$ as one of the many possible partitions of the set $X$. Figure 3.1 illustrates an example of two possible partitions for the same model. Given a set of feasible partitions, the task of choosing the partition that best describes the input dataset is hard, because it depends mainly on the type of information an application aims at exploring from the input dataset. In fact, this ill conditioning problem leads to a lack of consensus about the formal definition of a region and its homogeneity properties in segmentation context [UPH07, HJBJ+96].

Thus, multiple partitions can correspond to many different human interpretations of a surface, with different region refinement. Figure 3.1 illustrates such perception grouping ambiguity, where, even though the results are quite different from each other, the produced regions by both segmentations correspond well to human perception.

In the absence of an unique ground-truth segmentation, the design of a segmentation algorithm is only relevant when done and measured in the context of the larger system into which it will be incorporated [Pan08]. We believe that segmentation algorithm used as a pre-processing step for registration should have the following crucial characteristics:

1. *Correctness.* The segmentation should produce regions where the level of detail extracted depends uniquely on the input dataset.

2. *Data-driven.* The segmentation algorithm should produce regions of arbitrary shape. No prior knowledge about the surface models composing the *scene*, neither the error model, should be required. In this way, the segmentation can be performed on a wide range of surfaces types, such as free-form and sculptured-based surfaces.

3. *Region homogeneity predicate.* All regions produced by segmentation must respect some pre-defined homogeneity predicate.

4. *Robustness to noise.* The algorithm must produce correct segmentation even when the input dataset is corrupted with noise.

5. *Stability with respect to parameter choice.* Only few control parameters should be necessary to tune the segmentation, and these parameters should have an intuitive and physical meaning. In addition, the algorithm must produce segmentations of consistent correctness for a range of parameter choices [UPH07].

6. *Scalability.* The algorithm must segment directly large datasets without any pre-processing, such as re-sampling and pre-clustering.

7. *Complexity.* Segmentation algorithms should run at speeds similar to the number of input points, up to some low constant factor [FH04]. This property is important to make an algorithm practical, once segmentation is only a pre-processing step.

If a segmentation algorithm satisfies these requirements, then it will give useful and predictable results. Consequently, the segmented regions can be reliably incorporated into larger systems. In our region-based registration framework, for example, we use regions to perform global registration between two *scans* that overlap partially. In this application, we implicitly rely on the regions generated by the raw point cloud segmentation to be "useful", which means that these regions should be large enough to compute higher-order statistics of *scene* structure, but small enough to fall within boundaries.

One very important consequence of using the characteristics introduced above to evaluate a segmentation algorithm is the fact that the correctness of a segmentation does not rely on the ability of the algorithm to generate a partition of the input dataset that mimics human

perception. Instead, a segmentation algorithm produces correct results when the assumptions made, and the homogeneity predicate defined by the segmentation generates regions having such properties. Finally, the robustness of the algorithm is evaluated in terms of the invariability of these results under variation on the input dataset.

## 3.3 Foundation of 3D sampled surface segmentation

We will present in this section some fundamental definitions and properties of any 3D segmentation algorithm. We propose a taxonomy that categorizes segmentation algorithm according to the regions they generate and the hypothesis they make about the input dataset. We end up the section reviewing the main works done in each category, and how they perform, when considering the characteristics introduced in section 3.2.

### 3.3.1 Segmentation definition

In order to help 3D segmentation algorithms design, comparison and evaluation task, Hoover *et al.* [HJBJ+96] presented a common formal 3D segmentation definition:

**Definition 3.1** *Given a set $X$ of samples, the segmentation process consists in generating a partition $\mathcal{P}$ of the set $X$, formed by the disjoint subsets $R_1, R_2, \ldots, R_n$ (regions), where the following properties hold:*

1. $R_1 \cup R_2 \cup \cdots \cup R_n = X$. *It means that each point belongs to an unique subset.*
2. *Every subset is spatially connected.*
3. $\forall (R_i, R_j) \in X \times X$, *with* $i \neq j$, $R_i \cap R_j = \emptyset$. *Regions do not overlap.*
4. $\forall R_i \in X, \bar{P}(R_i) = TRUE$. *All points in a subset satisfy the specified similarity predicate.*
5. $\forall (R_i, R_i) \in X \times X$, *with* $i \neq j$, *and* $R_i$ *and* $R_j$ *are adjacent,* $\bar{P}(R_i \cup R_j) = FALSE$. *If subsets are neighbors, they represent different regions.*

From the listed properties, $R_i$ is an open set and it is expected that an algorithm partitions the input sampled surface into connected non-overlapping regions satisfying some homogeneity predicate. These properties are the requirements needed only to the segmentation process. They do not restrict neither the region homogeneity predicate neither the geometry and shape of regions obtained from segmentation. It is mainly why different algorithms generate different segmentation results, even when they verify the segmentation properties of definition 3.1.

### 3.3.2 A taxonomy for 3D segmentation techniques

Most of segmentation algorithms applied to 3D datasets are adaptation of methods designed to segment 2D images [BFL06]. What differentiate one approach from another are the specific criterion function $J : (2^{|X|} - 1) \rightarrow \mathbb{R}$, which is a function of the partitioning of $X$ to be optimized,

and the technique used to this optimization. Once they are defined, the segmentation problem is one of finding the partition $\mathcal{P}$ that extremes the criterion function. Different criterion functions, and optimization techniques produce regions with different properties, like geometry, shape and detail level.

We propose a 3D segmentation algorithm taxonomy which aims at characterizing the many existing 3D segmentation algorithms according to criterion function to be optimized and the properties of the regions they generate. This taxonomy groups techniques that produce equivalent segmentation. It guides the choice of the evaluation and comparison criteria to analyze segmentation algorithms in the same category. Algorithms at the same level but from different categories are unrelated to each other, and they cannot be compared objectively and quantitatively because they do not generate equivalent regions. Figure 3.2 shows the 3D segmentation taxonomy. We will examine each category, with emphasis on the categories our algorithm falls into.

Any segmentation algorithm falls into one of two general classes: contour-based and region-based methods. Since, whatever the chosen approach, the segmented data will contain both region and contour information, hybrid techniques have also been developed [KS00]. Contour-based or edge-based methods formulate segmentation as an attempt to extract regions by locating points that lie on their boundary. A contour is a closed curve, either continuous or discrete. The output of contour-based segmentation algorithms is a set of closed contours and a region is then defined as the set of points enclosed by a contour.

A very different segmentation formulation is used by region-based methods. They formulate segmentation as a gathering of similar feature points in order to form regions. The criterion function to be optimized is a region homogeneity predicate and a region is defined as a the biggest subset that verify such homogeneity predicate in the feature space. The output of methods in this category is a set of homogeneous regions.

To be more precise about the region homogeneity predicate, we break region-based algorithms into two categories: model-driven and data-driven methods. Model-driven methods can be seen as a surface fitting problem, where points are fit to some surface model and the criterion function to be optimized is the fitting error. These methods are more or less application-dependent, since they rely on prior knowledge of the elementary surface models composing a *scene*. The methods in this category differentiate from each other by the types and the parametrization of the surface models and by the strategy used to assign a point to a given model. Generally, segmentation is performed into two stages. The first stage, also called surface classification, assigns to small subsets a surface model, from the possible surface models set, that minimizes the fitting error. In the second stage is performed the surface parameter estimation. This is done by fitting the surface model to the set of inliers that actually describe such surface. The output of model-driven methods is a region set, where each region is represented by a parametric surface and by the list of inliers.

Data-driven methods, on the other hand, use concepts and techniques from differential geometry to describe shapes of arbitrary smooth surfaces arising in 3D datasets. They group points by evaluating some local homogeneity predicate, based on local feature analysis, and

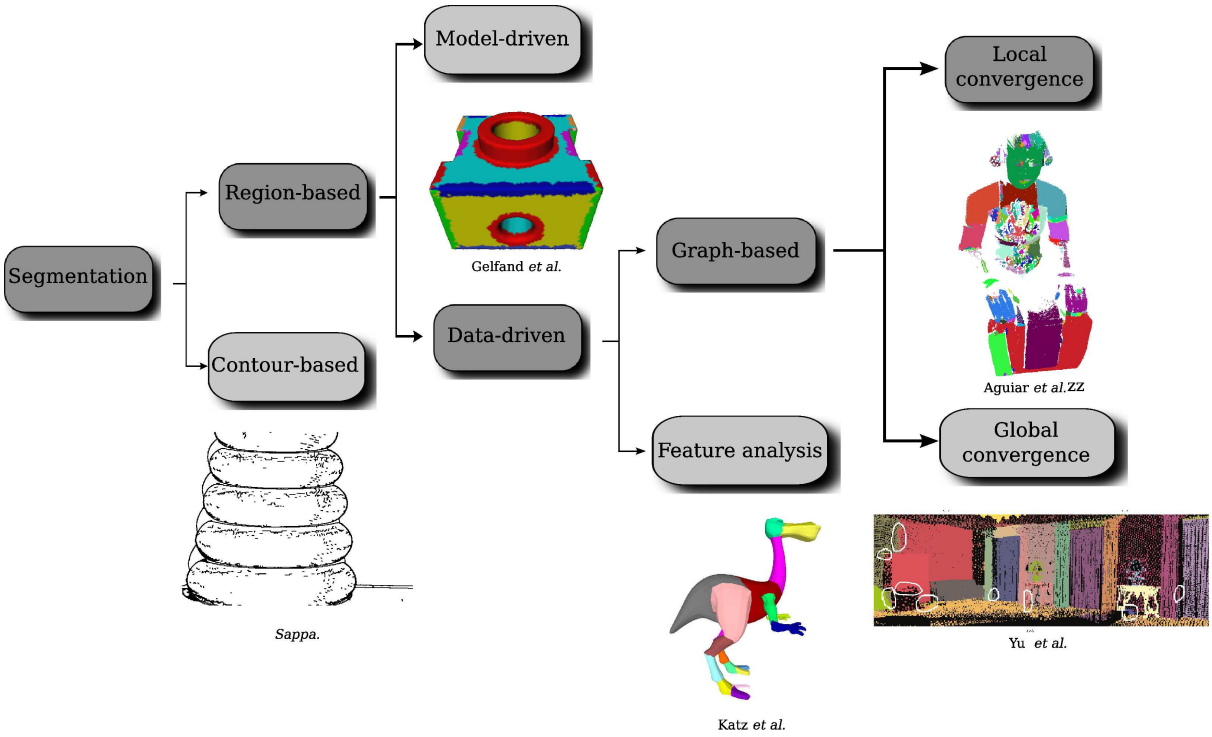



**Figure 3.2:** *3D segmentation taxonomy.*

combine local shape properties into global ones. Difficulties arise when local descriptors are used to identify the shape and global similarities of arbitrary surfaces, because the mathematics of differential geometry gives little guidance for an integrated global shape description [BJ88]. Data-driven segmentation algorithms differentiate from each other by the way in which points are gathered into regions and their region homogeneity predicate. If we consider the latest, or the criterion function, data-driven segmentation algorithms can be either feature analysis based or graph-based.

Data-driven, feature analysis based approaches, form regions that are "flat" in the feature space. Dense regions in the feature space thus correspond to local maxima of the p.d.f. [CM02]. Once the location of a local maxima is determined, the region is delineated based on the local structure of the feature space. The criterion function to be minimized is generally the feature variance within a region or some other related error measure.

Data-driven, graph-based approaches, on the other hand, form regions where the criterion function to be optimized is related to the graph *minimum cut* [SM00]. Here, the graph explicits points proximity information and the edges weight the similarity in some feature space. Together, they defines the search space of all feasible partitions.

Finding the *minimum cut* in a graph is a NP-hard computational problem. To solve the segmentation problem as a graph bi-partition, two different strategies have been proposed. First, there are the approaches that guarantee the convergence of the criterion function towards the global minimum. The graph bi-partition is done by finding the optimal *cut* whose cost is minimum among all possible graph *cuts*. The other graph-based segmentation category, instead of searching for the *minimum cut* to bi-partition the graph, aims at decreasing the search space

of possible partitions to a subset of more likely partitions and perform segmentation under boundary evidence. Boundary evidence corresponds to areas where there is a strong change of the feature value. These methods converge towards the local minimum of the criterion function and they do not explicitly evaluate the *cut cost* to perform segmentation.

This 3D segmentation taxonomy can still be refined if we consider the approximation technique used to reach a solution. A segmentation algorithm in any category can be classified into bottom-up and top-down. Bottom-up, or region growing, approaches begin with $n$ regions, each region containing exactly one point. Regions are constructed from seed points and the number of seeds determine the maximum size of the region set. Then, neighbor regions are recursively merged thanks to a similarity function. As regions are merged, the seed points move toward the center of the region. The segmentation stops when the homogeneity predicate is no longer verified on the neighborhood of the region. The seed choice is the most critical step of bottom-up approaches, because seeds placed on noisy or boundary points lead to erroneous results [KS00]. Most of methods falling in this category propose algorithms to robustly place these seed points. In top-down approaches, the original dataset is initially represented by an unique region. This region is recursively subdivided into smaller ones until each region is connected and respects the homogeneity predicate.

In the next section, we will examine the main works done in each category in the perspective of using them to raw point cloud segmentation. The evaluation criteria rely on the characteristics presented in section 3.2.

### 3.3.3 Review of 3D segmentation techniques

Limited work has been proposed in the literature to solve the problem of raw point cloud segmentation [ADC07b, YFM01]. The difficulty of performing segmentation directly on raw point clouds from *scans* is that the input data are ill-conditioned. As discussed in Chapter 2, raw point clouds are not regularly sampled in the 3D space, and artifacts are pretty common. The objects represented by the point clouds can take an enormous number of shapes and they should be represented by various families of surface models which have different parametrization dimensions. Additionally, raw point clouds from *scans* provide only partial object information, and neither the surface models composing the object nor the object's topology are known *a priori*. Although many segmentation algorithms have been proposed in the literature to deal with both 2D images [SM00, CM02, Wei99, DV06] and 3D datasets [BJ88, YL89, Tau91, MPB04, NG04, KS00, MW99, CF01, JM07, ZG08, FB04, HTZ04, LCS06, GKH08], their use to unstructured point cloud segmentation is not straightforward. In this section we briefly consider some of the 3D segmentation related work of each category and in the next section we will examine in detail the graph-based data-driven segmentation methods.

#### Contour-based segmentation algorithms

Contour-based segmentation algorithms have been extensively used to 3D sampled surface segmentation [AM96, BS02, Jia00, MB05, BFL06]. An example of contour based segmentation algorithm is presented by Sappa in [Sap06]. Sappa proposed a contour-based segmentation

algorithm that extracts closed contours from range images. Sappa uses an unsupervised graph-based and the $MST$ is the support to closure refinement. Given a set of edge points, the $MST$ covering this dataset suffers iteratively morphological operations of dilatation, erosion, followed by branches pruning, until it obtained only closed contours. The algorithm is carried out with no prior information about the surface shape, and it takes as input the points edges, which is obtained from segmentation.

#### Model-driven segmentation algorithms

A large number of model-driven segmentation methods can be found in the literature [Bes88, BJ88, Tau91, BMG94, KS00, GBBS04, NG04, GKH08]. The segmentation algorithms in this category are mostly used in reverse engineering applications, which deals with reconstructing a CAD model from an unstructured input dataset. If a *scene* is composed only by elementary surface models, like, for example, planes, spheres, cones, and surfaces of revolution, then the segmentation ground truth is usually unambiguously defined. If a model-driven approach incorporates such surface models, it is then expected that both regions and edges are accurately extracted and the use of a model-driven approach is justified. When such assumptions are verified, the evaluation of the segmentation performance can be done quantitatively. In [HJBJ$^+$96], a good overview of a model-driven range image segmentation algorithms is presented, and they propose a comparative framework to evaluate segmentation results, when it is provided the ground truth segmentation. Experimental results are provided to compare the methods presented. Bowyer *et al.* [JBM$^+$00, MPB04] refined the region definition made in [HJBJ$^+$96], and they present a comparative framework, where, besides planar regions, curve regions are also accepted.

From the many proposed model-driven segmentation algorithms, Han *et al.* [HTZ04] is the only one, to our knowledge, that proposes to segment noisy *scans* from natural *scenes*. They use both geometric and reflectance information, and they partition the input dataset into a large number of surface models, like planes, conics, splines and 3D histogram for nonparametric free-form objects. They formulate the segmentation in the Bayesian framework, where the posterior probability is distributed over a search space with a countable number of subspaces. Each subspace is a combination of surface models. To explore the search space, the algorithm simulates both reversible jumps and stochastic diffusion, which, in combination, simulates a Markov chain process sampling from the Bayesian posterior probability. Even though their approach is robust to noise, without reflectance information, the segmentation using only geometric clues performs poorly.

#### Data-driven based on feature analysis segmentation algorithms

Most of the algorithms in this category perform segmentation on triangle meshes, where the main objective is to extract semantic information from the input dataset. One class of feature analysis based methods assume an uniform-sampling, smooth input data which represents the entire object, or a watertight 3D digital model. Then, different topological structures, like medial axes [DGG03] and Reeb graphs [MP02, HSKK01], are used for shape segmentation. None of

such approaches can be used in raw point cloud segmentation, since the watertight hypothesis is never verified.

The second class of feature analysis based methods perform a preliminary clustering of local shape descriptors associated with points proximity. The features considered must be invariant under rigid transformation and the regions are homogeneous in the feature space. In [HP05] a top-down segmentation algorithm generates regions with homogeneous integral volume descriptor [PWHY09]. In [Pau03] the noisy point cloud is recursively split along the direction of greatest variation defined by the split plane composed by the centroid and the second largest eigenvector of the covariance matrix. In [YLL+05] the mean shift segmentation algorithm, which is a non-parametric clustering algorithm [CM02], is used to partition a triangle meshes, where the feature space is a 6D points, composed by triangle centroids equipped with triangle normals.

## 3.4 Graph-based segmentation

Considering the problem of raw point cloud segmentation, it quickly becomes clear that some segmentation categories cannot be used to solve the problem. While contour-based approaches may not extract correctly the contour of *scans*, since *scans* represent surfaces with boundaries, all model-driven segmentation algorithms proposed in the literature, to our knowledge, are unable to efficiently and robustly partition correctly sampled surfaces into a larger type of surfaces models, such as the ones found in free-form and man-made objects. The use of a data-driven segmentation algorithm seems to be the logical choice to solve the problem of partitioning raw point clouds.

Among data-driven approaches, the use of a feature analysis approach is not the most appropriated, because it relies on the analysis of the feature space, in which features are mostly estimated from the input data. Graph-based approaches, on the other hand, appears to be the most appropriate choice, since the criterion function to be minimized is the graph *cut*, not the feature value itself. Graph-based approaches generate regions of arbitrary shape, they have only a few input parameters, that are stable, with intuitive tuning, and they produce an arbitrary number of regions, which shape depends mainly on the input data.

Our segmentation algorithm falls in the graph-based segmentation category. In this section we present the definition and general properties of graph-based segmentation methods and we examine some methods falling in this category and which are designed to partition 3D sampled surfaces.

### 3.4.1 Background on graph-based segmentation

First, we will introduce some terminology. Let $G = (V, E, W)$ be a graph. The nodes in $V$ correspond to data points, edges represent neighborhood relationships, and edge weight reflects similarity between pairs of linked nodes. The weighting function $w : V \times V \rightarrow \mathbb{R}^+$ assigns the similarity between samples in some feature space. The graph embeds the relationships between samples $\mathbf{x}_i \in X$ and it explicits similarity between them $(\mathbf{x}_i, \mathbf{x}_j) \in E$ through the weight $w_{i,j} \in W$. Based on these considerations, the graph-based segmentation can be stated

as:

**Definition 3.2** *A graph $G$ can be partitioned into two disjoint sets, $R_1$, $R_2$ by simply removing the edges that connect nodes from $R_1$ to nodes in $R_2$. This graph bi-partition defines a segmentation according to definition 3.1. The degree of dissimilarity between these two regions can be computed as the total weight of the edges that have been removed also called the cut, which is defined as:*

$$\eth(R_1, R_2) = \{(\mathbf{x}_a, \mathbf{x}_b) \in E \mid \mathbf{x}_a \in R_1 \wedge \mathbf{x}_b \in R_2\} \tag{3.3}$$

*Given a cut, the cut value is given by:*

$$cut(R_1, R_2) = \sum_{\mathbf{x}_i \in R_1, \mathbf{x}_j \in R_2} w_{i,j} \tag{3.4}$$

*The optimal bi-partition of a graph is the one that minimizes this cut cost, when the weight $w_{i,j}$ embeds similarity between points. The partition criteria sought in graph-based segmentation algorithms are to minimize the disassociation between regions and to maximize the association within the regions [SM00].*

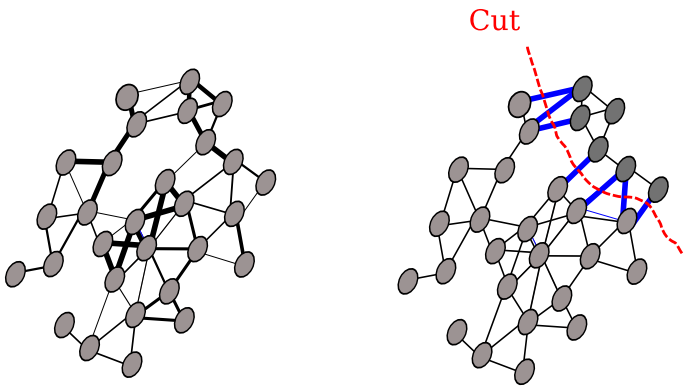



**Figure 3.3:** *A 2D graph-based segmentation example. The cost of each edge is reflected by the edge's thickness. A globally optimal segmentation is the graph bipartition that minimizes this cut cost.*

Figure 3.3 illustrates the principle of graph partition. The analysis of the complete solution space to find the optimum graph *cut* is a NP-hard computational problem. Although there is an exponential number of such partitions, finding the *minimum cut* of a graph is a well-studied problem and there exists efficient algorithms for solving it [SM00, BK04]. An alternative to this problem is to partition the graph under boundary evidence, without explicitly computing the *cut*, and this graph partition formulation was adopted by some graph-based segmentation algorithms [CBV98, JM07]. Note, in Figure 3.3, that, from the graph-based segmentation definition, the graph bipartition can generate regions with different shape and size. In order to minimize the unnatural bias for partitioning out small sets, caused by the *cut* criterion, other partition criteria have been proposed in the literature [SM00, BK04].

Given a feature space $I$ and a similarity function $w$, a graph-based segmentation algorithms respect the following properties:

- Each region $R_i$ corresponds to a connected component in a graph $G'' = (V, E')$, where $E' \subseteq E$. In other words, any segmentation is induced by a subset of edges in $E$.
- Region shape is unknown *a priori* and the shape is defined by the feature space, the distance used to weight the graph edges, and the segmentation input parameters.
- Given a feature space and a similarity function, a region is characterized by a limited feature variation.
- Intrinsically, graph-based segmentation approaches partition the graph by evaluating whether or not there is evidence for a boundary between two regions.

These graph-based segmentation properties correspond well with the segmentation requirements established in section 3.2. Graph-based segmentation methods can be distinguished by the type of criterion function they optimize and by the technique for minimizing it. Although there is a large number of graph-based approaches proposed to solve the segmentation problem, we discuss next the main works done in each category.

### 3.4.2 Review of graph-based 3D segmentation algorithms

Over the past decades, the problem of 3D segmentation using a graph-based approach has received significant attention and numerous algorithms have been proposed to solve the problem. Most of them, however, were designed to partition watertight, smooth, and uniform meshes. Only a few algorithms handle noisy, unstructured, non-uniform point cloud with holes and boundaries. We will present the most popular N-D graph-based segmentation algorithms. Each of the algorithms has different strengths and weaknesses which we will briefly describe here in the context of raw point cloud segmentation. A summary of the algorithm's properties is showed in table 3.1.

#### Global convergence methods

There are many graph-based segmentation algorithms that fall in this category [SM00, BK04, BFL06]. Although some of them have formulated the segmentation to a N-D images [BK04, BFL06], there has been little work done in 3D sampled surface segmentation. The only graph-based segmentation algorithm that converges toward the global minimum proposed to 3D sampled surface segmentation is the normalized cut algorithm.

Yu *et al.* [YFM01] adapted the normalized cut segmentation algorithm, introduced in [SM00] to 2D image segmentation, to partition large unstructured dataset. The normalized cut, or simply Ncut, segmentation algorithm is widely used in 2D image segmentation, where, for a given graph bi-partition, the minimal cut can be found as a generalized eigenvalue system, originated from spectral graph theory. Its well established theoretical basis guarantees the convergence towards the global minimum of the cost function.

The use of graph-cuts in 3D image segmentation is still limited, due to mainly their high computational complexity ($O(N^2)$). To overcome this problem, Yu proposed to perform the

| | *Ncut* [YFM01] | Watershed and variants [MW99, JM07] | MST-based [ADC07b] | Image Foresting Transform [FSdAL04] |
|---|---|---|---|---|
| **Shape descriptor** | Any feature space | Features invariant to rigid transformation | Any feature space | Any affine feature space |
| **Post processing overhead** | Boarder and region refinement | Region merging | No | User interaction |
| **Scalability** | No | Yes | Yes | Yes |
| **Robustness to noise** | Yes | No | Yes | Yes |
| **Robustness to non-uniform sampling** | Yes | Yes | Yes | Yes |
| **Number of regions** | User-defined | User-defined | Undefined | User defined |
| **Convergence** | Global | Local | Local | Local |

**Table 3.1:** *Summary of graph-based segmentation algorithms that partition 3D sampled surfaces.*

*Ncut* segmentation after a pre-processing clustering stage. The clustering stage reduces data volume up to 40.000 points and it consists in grouping points with similar normal vector and color variation. After segmentation, a post-processing stage refines both boundaries and regions.

In [YFM01], the graph edges consist of both local neighborhood and random long-range connections to enforce global context. The edges weight is a mixte distance that takes into account both Euclidean distance, angular and intensity similarity between points, where the last two are in the form of a Gaussian distribution. The output regions have unknown shapes with smooth color and normal variation. One drawback of this graph configuration is the bias caused by mixte distance, which may reduce the effectiveness of each feature space on the resulting segmentation. Yu supposes a local Gaussian distribution for both feature spaces, which is not true when working on the data coming directly from the range system [BMG94]. Even if the addition of random long-range connections enforces global context, the vertices configuration also adds to the search space of possible partitions some regions that correspond to non compact surfaces.

The normalized cut cost function favors the partition of the input dataset into regions with similar size, or regularly-shaped regions, and this behavior obligates a too fine segmentation to extract small features. The fact that the number of regions to create is an input parameter in *Ncut* algorithm is a strong restriction to incorporate this algorithm into a larger system. Finally, the necessity of both pre and post-processing in order to treat large amount of data, including user intervention, is not practical and interferes with the algorithm performance.

### Local convergence methods

The merits of the graph-based segmentation algorithms that converge to the local minimum of criterion function are that they run in time linear in the number of graph edges, they perform segmentation directly on large datasets, and their results have acceptable global properties. These methods do not explicitly evaluate the *cut cost* to perform segmentation but they evaluate local graph properties to find boundary evidence. Algorithms in this category are among the standard segmentation algorithms and there are large amount of algorithms proposed in the literature [UPH07, CBV98, MW99, PKA03, APFJ$^+$08a, JM07, BJ88, ZY96, AB94, KS00]. We focus on works done to perform segmentation of 3D sampled surfaces.

**Watershed segmentation and variants [MW99, PKA03, APFJ$^+$08a, JM07].** Originally designed to 2D image segmentation [BM93, RM00], the watershed segmentation algorithm has become one of the standard techniques to partition triangle mesh surfaces. Mangan and Whitaker [MW99] proposed a graph-morphology based, bottom-up, algorithm to partition triangle mesh surfaces. There is no explicit energy function to be minimized and they use boundary evidence to form regions. The original algorithm is designed to provide good segmentations only for uniform meshes. In [JM07] is presented a variation of [MW99], where it is taking into account non uniform sampling and they proposed an adaptive threshold selection technique.

The variations of the watershed algorithm [PKA03, JM07] could be directly used to raw point clouds segmentation by replacing the *neighborhood graph* over the triangle mesh to structure the dataset. However, the watershed segmentation algorithm is sensitive to the edges configuration, since they are used to identify local curvature minima, and even small variations on the neighborhood size can result in completely different segmentations. Even if a post-processing region merging avoids too fine segmentation, the watershed segmentation algorithm is quite sensitive to noise because it is restricted to features invariant under rigid transformation, like curvature and other local shape descriptor, which are not robustly estimated in raw point clouds. Besides the watershed algorithm performs poorly for "textured" surfaces such as the Stanford bunny and dragon.

**The Image Foresting Transform (IFT) [FSdAL04].** Falcão *et al.* [FSdAL04] presented a region growing segmentation algorithm to N-D images that computes shortest-paths in a weighted graph to find region boundaries.

The use of shortest-path in vision problems has been extensively studied [Mon71]. Their main contribution is to formalize the shortest-path approach to allow the construction of spanning forests of multiple sources, which is essentially Dijkstra's procedure for computing minimum-cost paths from multiple sources in a graph [Dij59]. They proved that IFT and watershed segmentation algorithms were equivalent. They also proved the correctness of their algorithm to smooth path-cost function, and they set the hypothesis, the limitations and the extensions of their algorithm. Although they formulated the *IFT* segmentation to N-D images, there is not any work done on 3D sampled surface. Like any region growing approach, the IFT segmentation algorithm rely on the accurate choice of seed points. In the original work, this problem was not

treated and the seed were taken manually. Although there are various ways to pick these seed points automatically [BJ88, ZY96, AB94, KS00], this issue becomes critical in raw point clouds segmentation. This is because of the presence of noise and non-uniform sampling that affects both feature estimation and graph connectivity.

**MST-based raw point cloud segmentation algorithm [ADC07b].** Our graph-based approach to 3D raw point cloud segmentation belongs to the group of methods that uses the minimum spanning tree ($MST$) to decrease the search space of all possible partitions and the graph bi-partition is done under boundary evidence. In [FH04] and [XOU96] the original clustering algorithm is adapted to 2D image segmentation. Their experimental results showed that, although this method converges to a local minimum, it produces segmentation with satisfactory global properties.

The $MST$-based is efficient because the search space is restricted to $MST$ edges. Noise changes the edges configuration on the $MST$ and thus, the search space of possible partitions, even if it is variant under rigid transformation. Because of its high frequency profile, noise is explicitly identified on the $MST$, which prevents too fine segmentation. Surface roughness can be seen as region with high, but homogeneous, edge variation. The use of an adaptive threshold allows to correctly extract such surfaces. Compared to the other graph-based segmentation algorithms, the $MST$-based has some advantages. First, the segmentation can be performed in any feature space. Since the normal vector is quite accurately estimated in noisy point clouds, it can be taken as feature, even if it is variant under rigid transformations. Second, noise and surface roughness are both explicitly taken into account during segmentation. Finally, like the other graph-based segmentation algorithms that converge towards the local criterion function, the $MST$-based algorithm is applicable to large datasets. In the following sections, the $MST$-based segmentation algorithm is further discussed.

## 3.5 Graph based segmentation of dense, unstructured, 3D point cloud

In this section we will present a new graph-based segmentation algorithm that produces regions that verify the region property requirements, introduced in section 3.2. Our work is an extension of segmentation algorithms using the $MST$, introduced in [Zah71a] and later improved in [FH04]. The $MST$ is a tree of the original graph, where, in a noise-free case, the *paths* of the $MST$ traverse the entire input dataset passing on the isolevel of feature space. The segmentation procedure consists on traversing the edges in $MST$ looking for boundary evidence. The solution space of feasible partitions is given by the $MST$ edges configuration. This segmentation definition is coherent since any feasible *cut* either passes near a isolevel of the feature space either passes on the boundary between two regions. We define the initial search space of possible partition as the *neighborhood graph* $G_{\mathcal{N}}$ and the feasible search space of possible graph partitions is reduced by exploring the $MST$ of $G_{\mathcal{N}}$.

### 3.5.1 Overview of the algorithm

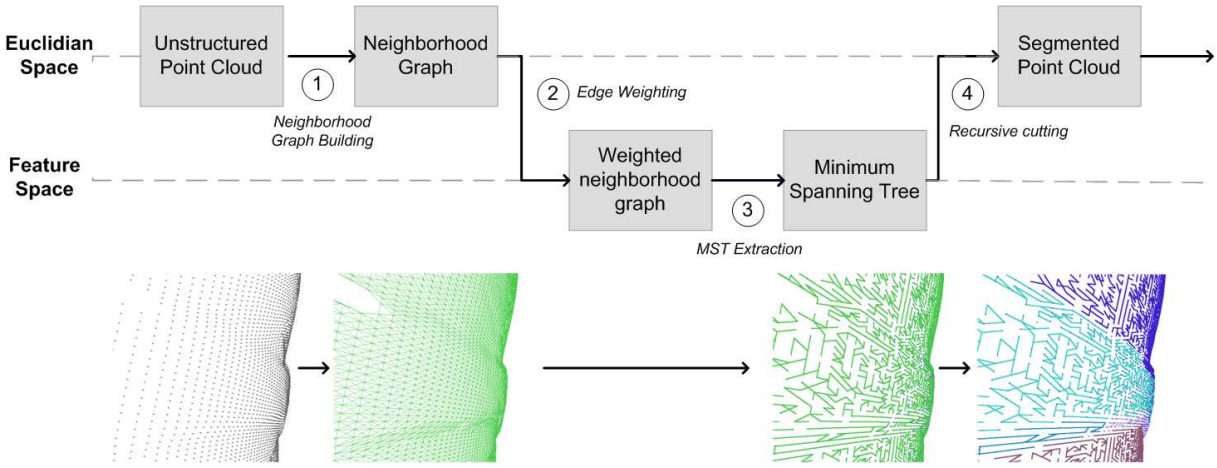



**Figure 3.4:** *MST-based segmentation pipeline. From a raw unstructured point cloud to a set of connected subsets.*

We formulate 3D point cloud segmentation as a partition of the *neighborhood graph* $G_{\mathcal{N}}$ into nonempty subgraphs. The algorithm pipeline is illustrated in Figure 3.4. The input of our algorithm is an unstructured raw point cloud and our algorithm consists of the following steps:

1. *Euclidean neighborhood graph construction.* The *neighborhood graph*, $G_{\mathcal{N}}$, is constructed to limit the search space of all possible graph partitions to connected subsets in $\Sigma$.

2. *Edge Weighting.* For each point $\mathbf{x}_i \in X$, there exists a feature value $f(\mathbf{x}_i)$ and the edges weights measure the dissimilarity between two connected points, in the feature space. Any feature can be taken to characterize the point cloud and the edges weights embed the variation of the local shape descriptors along the surface $\Sigma$.

3. *MST Extraction.* The segmentation is based on the analysis of the $MST$ of $G_{\mathcal{N}}$. The $MST$ is a sugbraph of $G_{\mathcal{N}}$ connecting all nodes in $G_{\mathcal{N}}$. There are two important $MST$ properties exploited on the segmentation. First, the $MST$ is the tree covering the entire dataset with minimal cost. It means that any edge $(\mathbf{x}_i, \mathbf{x}_j) \in MST$ is aligned, or points towards, the isolevel of $I$ at $\mathbf{x}_i, \mathbf{x}_j$. Second, any edge $(\mathbf{x}_i, \mathbf{x}_j) \in MST$ defines uniquely one *cut* on the *neighborhood graph*, and consequently, it defines one possible partition.

4. *Recursive cutting.* The solution space of feasible partitions is composed by the *cuts* defined by the edges $(\mathbf{x}_i, \mathbf{x}_j) \in MST$. The $MST$ is recursively traversed, and, at each iteration, the edge $(\mathbf{x}_i, \mathbf{x}_j) \in MST$ with $w_{\max}$, likely to be on the regions boundaries, is evaluated. A defined homogeneity predicate must be hold false in order to bi-partition the graph in $(\mathbf{x}_i, \mathbf{x}_j)$. The segmentation stops when all regions verify the homogeneity predicate.

Each step of the algorithm has different assumptions, complexity, and different strengths and drawbacks. In the following sections, we will detail each of these steps and their role in the segmentation.

### 3.5.2 Invariance of the MST through graph bipartition

The minimum spanning tree was an early approach in many computer graphics applications, such as clustering [Zah71a], 2D image segmentation [FH04] and feature lines extraction [PKKG03]. We will present the definition and properties of the $MST$-based segmentation algorithm and how it reduces the search space of all possible partitions to a smaller subset of more likely partitions.

#### Definition

Consider a weighted graph $G = (V, E, W)$. The minimum spanning tree $MST$, defined by the subgraph $\Gamma_G \subset G$, is a tree subgraph covering the entire dataset and $\Gamma_G$ is such that the sum of edge weights is minimum. As a skeleton of a data set, the $MST$ has some particular properties. First, $\Gamma_G$ is a sparse graph, since it has only $N-1$ edges, for $N$ nodes, and the *paths* $\tau \subset \Gamma_G$ form no cycles. Second, the tree with *minimal cost* can be seen as a set of *paths* that remain aligned with, or point towards, the closest isolevel of $I$, where $I$ is the feature space. Such properties are illustrated in Figure 3.5. In this example, the feature space is the $z$ coordinate and the similarity function is set as the Euclidean distance between points $z$ coordinate. Observe how the $MST$ *paths* are aligned with the isolevel of the $z$ coordinate.

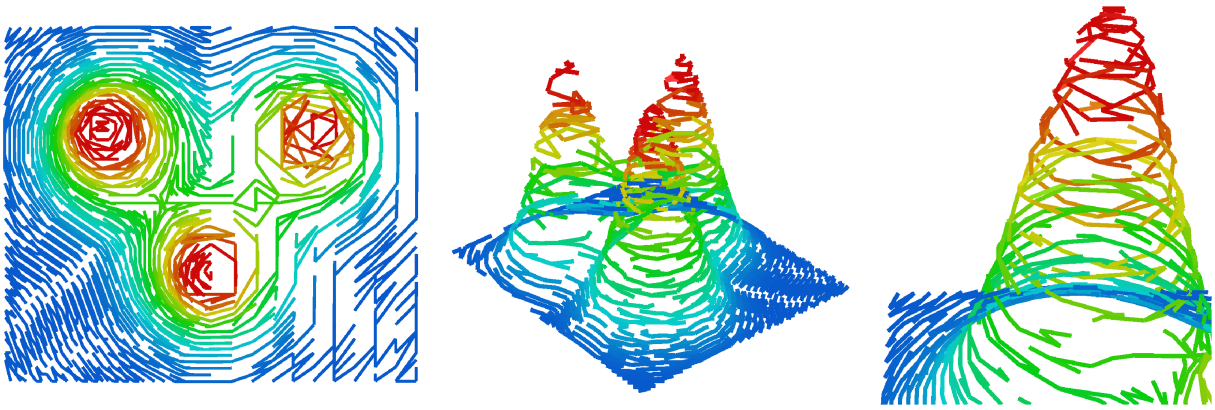


**Figure 3.5:** *The Minimal spanning tree is constructed by trying to connect points in order to remain in a same isolevel of $I$. In this example, the feature space is the $z$ coordinate and the similarity function $w_{i,j}$ is set as the Euclidean distance between points $z$ coordinate.*

From the $MST$ minimal cost property, there is no guarantee of the uniqueness of the minimal spanning tree connecting points within $X$, since multiple *trees* of *minimal cost* can be found. As will be later discussed, an edge rearranging in $\Gamma_G$ will change the search space of feasible partitions and the segmentation is sensitive to this rearranging.

#### Fundamental property

One important $MST$ property is that any edge $(\mathbf{x}_i, \mathbf{x}_j) \in \Gamma_G$ determines a unique *cut* on $G_{\mathcal{N}}$, and, by removing this edge, it results in a partition of $G$ into two disjoint subsets with their associated $MST$s. The next theorem enunciates such property.

**Property 3.5** *Given a connected graph $G$ and its $MST$ $\Gamma_G$, by removing any edge $(\mathbf{x}_i, \mathbf{x}_j) \in \Gamma_G$,*

*it is generated two disjoint spanning trees, which are the MSTs of the subsets obtained by the partition of the original set.*

**Proof:** This property can be summarized as follows. Let $G' = (V', E', W')$ be a subgraph of $G$. All edges of the $\Gamma_G$ that link the nodes $V' \subset V$ also belong to $\Gamma_{G'}$. In other words,

$$(\mathbf{x}_i, \mathbf{x}_j) \in \Gamma_G \Rightarrow (\mathbf{x}_i, \mathbf{x}_j) \in \Gamma_{G'} \tag{3.6}$$

Equation, (3.6) can be written as:

$$(\mathbf{x}_i, \mathbf{x}_j) \notin \Gamma_{G'} \Rightarrow (\mathbf{x}_i, \mathbf{x}_j) \notin \Gamma_G \tag{3.7}$$

Let $\Gamma_{G'}$ be the $MST$ of the subgraph $G' = (V', E', W')$. $\mathcal{P}$ is the partition of $G'$ into two non-empty disjoint subsets $A \subset V'$ and $B \subset V'$, with $A \cup B = V'$ and $A \cap B = \emptyset$.

Let $(\mathbf{x}_a, \mathbf{x}_b)$ be an arbitrarily *crossing edge*, with $\mathbf{x}_a \in A$ and $\mathbf{x}_b \in B$. Suppose that $(\mathbf{x}_a, \mathbf{x}_b)$ is not the *minimal crossing edge* of the partition $\mathcal{P}$. In other words, suppose that

$$\exists (\mathbf{x}_a, \mathbf{x}_b) \notin \Gamma_{G'} / \begin{cases} \exists \mathbf{x}_i \in A \\ \exists \mathbf{x}_j \in B \\ w_{i,j} < w_{a,b} \end{cases} \tag{3.8}$$

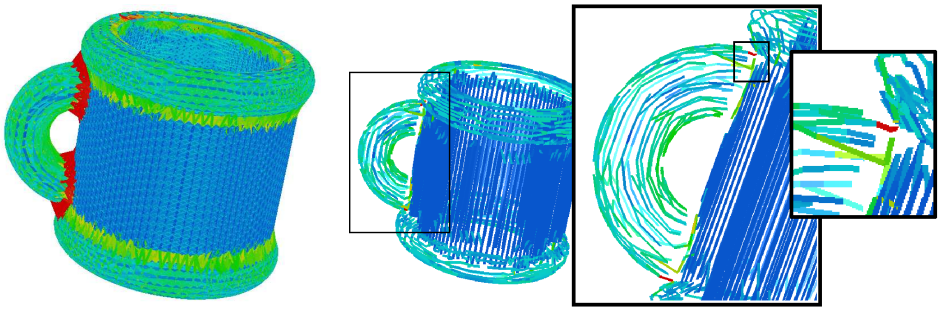
Now, consider the partition $\mathcal{Q}$ of $G$ into two non-empty disjoint subsets $A$ and $C = V \backslash A$, with $A \cup C = V$ and $A \cap C = \emptyset$. Since $E' \subseteq E$, we have that the edges $(\mathbf{x}_a, \mathbf{x}_b)$ and $(\mathbf{x}_i, \mathbf{x}_j)$ are also *crossing edges* of the partition $\mathcal{Q}$, where $\mathbf{x}_a, \mathbf{x}_i \in A$ and $\mathbf{x}_b, \mathbf{x}_j \in C$. Thus, the edge $(\mathbf{x}_a, \mathbf{x}_b)$ cannot be the *minimal crossing edge* of the partition $\mathcal{P}$, and, consequently, $(\mathbf{x}_a, \mathbf{x}_b) \notin \Gamma_G$, and the theorem is proved.

The $MST$ connectivity and minimum cost properties allow us to define the search space of the $MST$-based segmentation algorithm. Let $G$ be any graph and the search space of all possible partitions is given by $\{\mathcal{P}\}$. The search space of more likely partition is defined as the subset $\{\mathcal{P}'\} \subset \{\mathcal{P}\}$ obtained by removing edges $(\mathbf{x}_a, \mathbf{x}_b) \in \Gamma_G$. The size of this search space of feasible partition is reduced to $N - 1$, the number of edges in $\Gamma_G$, that is equal to the data size. Observe that this definition is independent of the datas dimension or the similarity function. Once the search space is established we will show how the segmentation is performed and the cost function we aim at minimizing. Once the search space is established we will show how the segmentation is performed and the cost function we aim at minimizing.

### Minimum spanning tree recursive cut

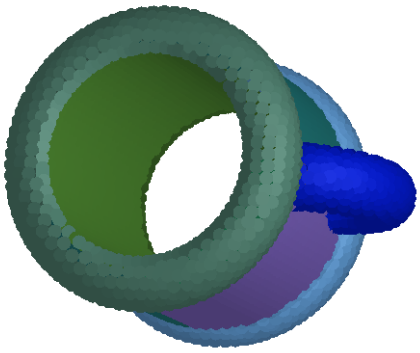
One property of the $MST$ is that the *paths* $\tau \in \Gamma_G$ traverse the entire input dataset passing on the isolevel of feature space. It means that edges between two vertices in the same component should have relatively low weights, and edges between vertices in different components should have higher weights. When one *path* passes from one isolevel to another, or from one region to

another, a larger edge weight variation is expected connecting these two parts. In a noise-free case, there is a unique edge in $\Gamma_G$ connecting two regions, or two isolevels. Thus, the basic assumption of a $MST$-based segmentation algorithm is that edges with large weight cost in $\Gamma_G$ are likely to be on a boundary between two regions. This situation is illustrated on Figure 3.6. Observe in Figure 3.6(a) that the edges weight within a region are similar and connecting two different regions have larger costs. In the $MST$ colormap (Figure 3.6 (b)) is observed that *paths* passing through edges with lower weight cost is preferred and there is only one edge, with large weight cost connecting two different regions. Segmentation removes such edges in order to form regions with homogeneous edges weight cost.



(a) *Neighborhood graph* colormap

(b) $MST$ colormap



(c) Final segmentation

**Figure 3.6:** *Segmentation through boundaries evidence.*

Let $R_i$ and $R_j$ be two neighbor regions obtained from a bipartition of $G$. From the subset of edges belonging to the *cut*, connecting points in the two distinct regions, we characterize the *cut* by the *minimal difference distance*, $Dif_{\min}(\eth(R_i, R_j))$, which is the smallest edge weight on the *cut*:

$$Dif_{\min}(\eth(R_i, R_j)) = \min_{(\mathbf{x}_a, \mathbf{x}_b) \in \eth(R_i, R_j)} w_{a,b} \tag{3.9}$$

Necessarily, the edge $(\mathbf{x}_a, \mathbf{x}_b) = \arg(Dif_{\min}(\eth(R_i, R_j)))$ must be in $\Gamma_{R_i \cup R_j}$ and it connects different regions, or different isolevels. If there is no edge connecting $R_i$ and $R_j$ we set $Dif_{\min}(\eth(R_i, R_j)) = \infty$.

Under such assumptions, the segmentation is performed by traversing the edges in $\Gamma_G$ looking

for boundary evidence. Graph partition is done by removing such edge when boundary evidence is confirmed. Such segmentation is coherent since any feasible *cut* $\mathcal{P}'$ passes near a isolevel of $I$ or passes on the boundary between two regions.

The segmentation based on boundary evidence takes into account only edges in $\Gamma_G$, not considering the *cut cost* between any potential partition. The $MST$-based segmentation is essentially the $MST$ extraction, using Prim's algorithm [Pri57], followed by its recursive traversal.

The algorithm produces a set of disjoint subgraphs, where the $MST$ of each subgraph verifies a homogeneity predicate. We will use an adaptive threshold as homogeneity predicate that compares the boundary evidence $Dif_{\min}(\supset(R_i, R_j))$ to the edges within the regions. This homogeneity predicate is adaptive with respect to the local characteristics of the data and produces homogeneous regions with bounded edge variation.

Although we use a top-down segmentation approach, the $MST$-based segmentation algorithm can be implemented in a bottom-up fashion. The last approach was adopted by Felzenszwalb and Huttenlocher in [FH04], where the segmentation is performed at the same time with the extraction of the $MST$. This approach is much more efficient because it traverses the entire dataset only once, while in top-down approach, the $MST$ of a region is traversed each search for boundary evidence.

#### MST reconfiguration caused by noise

Noise affects both feature estimation, graph connectivity, and, consequently, the $MST$ edges configuration. Therefore, an intrinsic property of the $MST$-based segmentation is that the search space of possible partitions is highly sensitive to noise. If segmentation is performed on raw point clouds without any pre-processing, such behavior is unavoidable.

The edge cost, $w$, is a differential function which amplifies the noise in a small area around the noisy point. Thus, the $MST$-based segmentation algorithm usually interprets noisy edges as a boundary evidence. However, noise has a small and limited range of influence on *neighborhood graph* weighting. It means that it is expected larger weight cost on edges leaving noisy points, compared to other nearby edges. As a consequence, these noisy points are the naturally placed in leaf nodes of the $MST$, since any *path* passing through noisy points neighbors have higher weight cost. We assume that noise does not change considerably the search space of partitions, since we perform segmentation on dense, large datasets, and the noise has only a local influence on the $MST$ configuration.

There are two possible strategies to make the $MST$-based segmentation algorithm robust to noise: build a pseudo $MST$, more robust to noise than the original $MST$, or filter noise during segmentation. The first strategy, proposed in [CZ05] and used in [Unn08], combine multiple $MST$s from the same graph in order to reduce their sensitivity to noise. They exploit a *perturbed minimum spanning trees (PMST)*, which is the $MST$ extracted from the point cloud, when this last is corrupted by a known noise model. Finally, the original $MST$ update the edges weight according to the frequency these edges appear in different $PMST$s. A second robust $MST$ construction, called *disjoint minimum spanning tree (DMST)*, is inspired on the Kruskal's minimum spanning tree algorithm and construct successively the graph $MST$, where

at each construction the edges from the previous $MST$ are removed on the original graph. At the end, it is manipulated a deterministic collection of $t$ $MST$s, where $t$ is the number of $MST$s. Both $MST$ construction strategies results in a pseudo $MST$ more robust to noise than the original $MST$. We say a pseudo $MST$ because the resulting tree is not necessarily the one minimal cost. However, the algorithms to construct robustly the $MST$ are both time and memory consuming, limiting their use to large datasets.

Our strategy to handle noise consists in using the original, noise affected, $MST$ and, during segmentation, separately process noise and boundary. From the property that noisy points are placed in the leafs of the $MST$, and it leads to the segmentation of the data into small and meaningless subsets. We solve this problem by setting a minimal region size. By using this strategy, we actually set the maximal noise influence on the $MST$ configuration, and we minimize the noise effect on the segmentation. In our approach, we set the minimal region size as proportional to the number of samples, and it is an user-defined input parameter.

### 3.5.3 Criterion function and homogeneity predicate

The $MST$-based segmentation algorithm is driven by the assumed knowledge of a homogeneity predicate $\bar{P}$. It identifies correct partitions by comparing boundary evidence to within regions feature variation. In this section we propose a homogeneity predicate $\bar{P}$ based on an adaptive threshold. The motivation for the threshold selection process described here is derived from [FH04] and [JM07].

Consider the regions extracted through $MST$-based segmentation. Such regions are obtained by removing some edges from the $MST$. Given a partition $\mathcal{P}' = \{R_1, \cdots, R_n\}$, each region $R_i \subset V$ is characterized by the *Maximal internal difference*, which is the largest edge weight in $\Gamma_{R_i}$:

$$Dif_{\max}(R_i) = \max_{(\mathbf{x}_a, \mathbf{x}_b) \in \Gamma_{R_i}} w_{a,b} \tag{3.10}$$

This measure relies on the properties of the tree connecting points within a region. The criterion function we want to optimize is the the *Maximal internal difference* within a region. The segmentation procedure is defined as the partition $\mathcal{P}$ that extremes this $Dif_{\max}(R_i)$. For this purpose, we define a region comparison predicate $\bar{P}$, which is an adaptive thresholding that evaluates the evidence of boundary between potential regions and feature homogeneity within regions. It is given by:

$$\bar{P}(R_i, R_j) = \begin{cases} \text{True} & \text{if } Dif_{\min}(\eth(R_i, R_j)) > \epsilon_{i,j} \\ \text{False} & \text{otherwise} \end{cases} \tag{3.11}$$

where $\epsilon_{i,j}$ is the threshold . In the standard $MST$-based segmentation, $\epsilon_{i,j}$ is a constant that can be either user defined either computed from the dataset. We use the adaptative threshold $\epsilon_{i,j}$, proposed in [FH04], which is defined as:

$$\epsilon_{i,j} = \min\left( Dif_{\max}(R_i) + \frac{\epsilon_{\min}}{|R_i|}, Dif_{\max}(R_j) + \frac{\epsilon_{\min}}{|R_j|} \right) \tag{3.12}$$

where $\epsilon_{i,j}$ is an adaptive data-driven parameter and it bounds the edges weight variation along the *paths* within $R_i$ and $R_j$. The parameter $\epsilon_{\min}$ controls the minimal region size and the degree $Dif_{\min}(\supset(R_i, R_j))$ must be greater than $Dif_{\max}(R_i)$ and $Dif_{\max}(R_j)$ so the region comparison predicate holds true and the partition is performed. Felzenszwalb and Huttenlocher in [FH04] have proved that the segmentation produced using an adaptive threshold obeys the properties of being neither too coarse nor too fine.

The parameter $\epsilon_{\min}$ is estimated as follows. Given the point cloud $X$, we compute a histogram of the edges weights. This histogram is an approximation of the probability density function (pdf) that represents the variation of the descriptor over the region. We choose $\epsilon_{\min}$ as the value on the p.d.f. function that guarantees to cover $K\%$ of the input edge set.

One very important property of the regions constructed under the predicate (3.11) is that these regions are homogeneous in the feature space. The consequence of such property is that regions can be equally homogeneous according to (3.11) but have different feature dynamic within the regions. Such behavior is illustrated in Figure 3.7. Although all regions respect the homogeneity predicate, the have distinct shapes and feature dynamics.

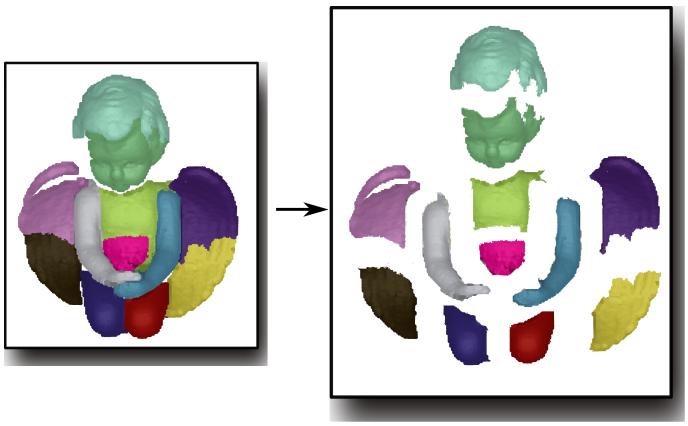


**Figure 3.7:** *Homogeneity predicate. Segmentation of an object using a fixed partition threshold with $\epsilon_{i,j} = \epsilon_{\min}$. Every region is homogeneous according to the edge weight cost, but different regions have distinct shape and feature distribution.*

### 3.5.4 Mathematical formulation of a region

In this section, we introduce a mathematical formulation of a region derived from the $MST$-based segmentation and we discuss its properties. We will use this region definition to prove that it respects the segmentation properties (definition 3.1) and it optimizes the criterion function (3.10).

#### Region definition and properties

We want to model the segmentation in terms of points connectivity and their similarity in some feature space. The $MST$-based segmentation algorithm can be done in any $N$-dimensional feature space. First, we define a region as a set of points connected by a singular *tree*, as follows:

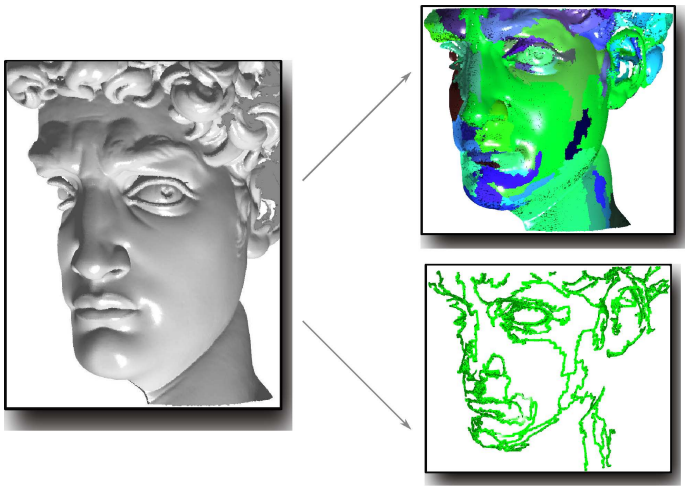


**Figure 3.8:** *Region and boundary definition. Given a unstructured raw point cloud, the segmentation produces a set of disjoint regions and boundaries between regions.*

**Definition 3.13** *Lets* $\mathbf{x}_0$ *be a point in* $X$*. The region* $R_i$

$$R_i = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_n\} \subseteq X \tag{3.14}$$

*is defined as the biggest subset of* $X$ *in which points within* $R_i$ *respect the following conditions:*

1. ***Seed point****:* $\mathbf{x}_0 \in R_i$ *is called seed point and it is the origin of any path* $\tau \subset \Gamma_{R_i}$*.*
2. ***Homogeneity****: any path* $\tau \subset \Gamma_{R_i}$ *must respect Euclidean distance constraint and feature homogeneity:*

$$\begin{aligned} &\forall \mathbf{x}_k \in X,\ \mathbf{x}_k \in R_i \Leftrightarrow \exists \tau = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k) \subseteq \Gamma_{R_i}, \\ &\quad with \quad \forall j \in [0, k-1] \begin{cases} \|\mathbf{x}_j - \mathbf{x}_{j+1}\| < r_i \\ w_{j,j+1} < \epsilon_i \end{cases} \end{aligned} \tag{3.15}$$

3. ***Connectivity****: an unique tree* $\Gamma_{R_i}$ *connects all points within* $R_i$*.*

The parameter $r_i$ incarnates the Euclidean spatial distance constraint between two points and this parameter is fundamental to guarantee that connected points in $R_i$ represent connected surfaces in $\Sigma$. The parameter $\epsilon_i$ characterizes the homogeneity predicate in feature space. This parameter limits the maximal magnitude of $w_{j,j+1}$ and it also sets the equivalence relation between points within a region. If, for example, we set $\epsilon_{j,j+1} \to 0$, a region will be "flat" in feature space. The control parameter $\epsilon_{j,j+1}$ can be either set constant [Zah71b] or adaptive [FH04].

This region definition also emphasizes a singular point that plays an important role for the region $R_i$: the seed point $\mathbf{x}_i$. In fact, this is only a consequence of our formalization. Actually, any point of the region can be taken as seed point. This is demonstrated in the following property.

**Property 3.16** *Let* $R_i \subset X$ *be a region. Any point* $\mathbf{x}_j \in R_i$ *can be taken as seed point of* $R_i$*.*

**Proof 3.17** *Let $R_i \subset X$ be a region. Suppose that $\mathbf{x}_0 \in R_i$ is the seed point of $R_i$. Given any two points $\mathbf{x}_a, \mathbf{x}_b \in R_i$, we have from definition (3.15) that*

$$\exists \tau = (\mathbf{x}_0, \ldots, \mathbf{x}_a) \subset \Gamma_{R_i}, \text{ such that } \forall j \in [0, a-1] \begin{cases} \|\mathbf{x}_j - \mathbf{x}_{j+1}\| < r_i \\ w_{j,j+1} < \epsilon_i \end{cases}$$

*and*

$$\exists \tau = (\mathbf{x}_0, \ldots, \mathbf{x}_b) \subset \Gamma_{R_i}, \text{ such that } \forall j \in [0, b-1] \begin{cases} \|\mathbf{x}_j - \mathbf{x}_{j+1}\| < r_i \\ w_{j,j+1} < \epsilon_i \end{cases}$$

*are hold true.*

*The path $(\mathbf{x}_a, \ldots, \mathbf{x}_b)$ can be expressed as the union of the two paths $(\mathbf{x}_a, \ldots, \mathbf{x}_0)$ and $(\mathbf{x}_0, \ldots, \mathbf{x}_b)$. Since the distances $\|\mathbf{x}_j - \mathbf{x}_{j+1}\|$ and $w_{j,j+1}$ are symmetric, we obtain*

$$\forall j \in [a, b-1] \begin{cases} \|\mathbf{x}_j - \mathbf{x}_{j+1}\| < r_i \\ w_{j,j+1} < \epsilon_i \end{cases}$$

*Hence, any point $\mathbf{x}_a \in R_i$ can be taken as seed point of $R_i$.*

**Do these regions define a segmentation ?**

In the previous section, we introduced a region definition based on the construction of *paths* connecting points within the region, and respecting some connectivity and homogeneity conditions. In this section we will show that the region defined in (3.15) actually forms a partition of $X$ and respects the segmentation properties defined in 3.1.

To prove that $\{R_i\}$ is a partition of $X$, we must show that:

1. $\forall R_i \in X, R_i \neq \emptyset$: because each region $R_i$ contains at least the seed point used for its construction, this property is respected.

2. $\bigcup R_i = X$: each point belongs to, at least, one region, since it can be used as a seed to build the region.

3. $\forall R_i, R_j \in X, R_i \cap R_j = \emptyset$: if two regions $R_i$ and $R_j$ have a non empty intersection, then at least one point $\mathbf{x}_0$ belongs to both of them. In this case, a path can be defined between any two points from $R_i$ and $R_j$ via $\mathbf{x}_0$ and this path would respect the constraints given in equation 3.15. Thus, $R_i$ and $R_j$ cannot be regions of $X$, because they violate the definition of region as the biggest subsets of $X$ respecting this constraint.

4. *Any region $R_i \subset X$ is a $r_i$-connected*: given any two points $\mathbf{x}_i$ and $\mathbf{x}_j$ in $R_i$. We can consider $\mathbf{x}_i$ as the seed point of $R_i$. Thus, by definition of the region, we can define a path inside $R_i$ between $\mathbf{x}_i$ and $\mathbf{x}_j$ where the distance between two consecutive points of the path is less than $r_i$.

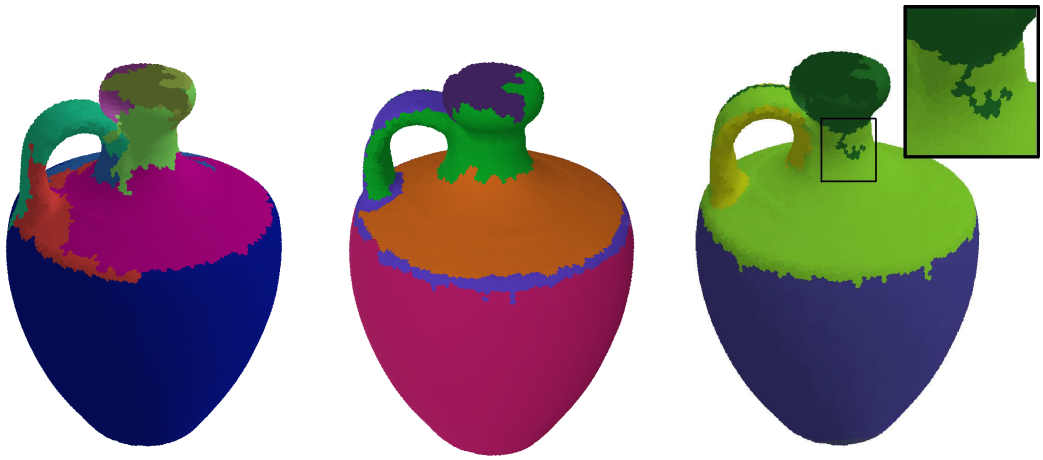
If we consider the properties of partition and $r_i$ connectivity of the regions, the definition given in 3.13 fulfills all the requirements given by Hoover to define a segmentation. We turn

our attention now to the problem of using the $MST$-based segmentation algorithm presented to partition dense, raw point clouds. We will then present how the graph is constructed from an unstructed dataset and how it is defined the feature space and the similarity measure between nodes.

### 3.5.5 Building the graph from raw point clouds

The input data to this problem is a 3D raw point cloud. In order to use the $MST$-based segmentation framework to partition this unstructured point cloud, we must represent the dataset as a undirected weighted graph. Besides points connectivity, other information are encoded in the graph, like feature and the similarity values, which must be defined. In this section, we will present how we build this graph, and which feature space and similarity functions are used. Once they are defined, we can infer the segmentation properties in terms of robustness to noise, number and size of regions.

#### Building the neighborhood graph



(a) $r = 1.0$ (b) $r = 3.0$ (c) $r = 6.0$

**Figure 3.9:** *Influence of the neighborhood size on segmentation. Example of segmentations when the only parameter that varies among these results is the neighborhood size. In this example, it was used the ball neighborhood system to build the neighborhood graph.*

Given an unstructured 3D point cloud $X$, the first step towards establishing the search space of all possible segmentations, through the graph $G$, is the definition of the edge set $E$. The edge set embeds points connectivity through points proximity in some feature space.

One common approach to build is based on mapping each point to some feature space, and then finding clusters of similar feature value. In this case, the graph $G = (V, E)$ has a node corresponding to each point and there is an edge $(\mathbf{x}_i, \mathbf{x}_j)$ connecting pairs of points $\mathbf{x}_i$ and $\mathbf{x}_j$ that are nearby in the feature space, rather than using neighboring points in Cartesian space. Segmentation then produces regions that are connected in feature space, but they do not necessarily represent compact surfaces in $\Sigma$.

We consider a construction approach. It consists in building the edge set by connecting neighboring points in Cartesian space and setting the edges weight as a similarity measure between points in some feature space. In this case, segmentation not only produces regions that respect some homogeneity predicate in feature space, but also these regions represent compact surfaces in $\Sigma$.

We assume that the point cloud is $\varepsilon$-dense and $\delta$-noisy, where $(\varepsilon + \delta)$ is estimated as proportional to the mean local sampling spacing over the input dataset. Although any symmetric neighborhood system can be taken, we choose the ball neighborhood system because it avoids the connection between points from different surfaces and between outliers. The ball neighborhood system is also preferred to restrict the search space to monolithic regions. To this end, the neighborhood size, given by the sphere radius $r$, is set proportional to $(\varepsilon + \delta)$, and we obtain a sparse graph. The neighborhood size determines the search space of all possible partitions, since the search space is proportional to the edge set size. Too small neighborhood size results in a small set of possible segmentations, but it restraints local homogeneity analysis to a small area, which makes the algorithm sensitive to noise. Too big neighborhood size, oppositely, enforces global information on the graph, is more robust to noise. However, too big neighborhood size also increases the number of possible segmentations, and it adds to the search space some non-monolithic regions as possible solutions. Figure 3.9 shows some segmentations when the only parameter that varies among these results is the neighborhood size.

### Feature space and dissimilarity function

(a) (b) (c)

**Figure 3.10:** *Example of segmentations in different feature spaces. Points within a region are homogeneous according to (a) normal variation (c) Gaussian curvature variation.*

The weighting function $w : X \times X \to \mathbb{R}^+$ is used to edge weighting $(\mathbf{x}_i, \mathbf{x}_j) \in E$ and it determines the feature space in which the segmentation is performed. The weight $w_{i,j}$ over an edge $(\mathbf{x}_i, \mathbf{x}_j)$ is a local measure of how likely the points are to belong to the same region. For a normalized dissimilarity function, when the weight cost $w_{i,j}$ is close to 0, the points $\mathbf{x}_i$ and $\mathbf{x}_j$ are likely to belong together, since they are close to each other in the feature space. On the other hand, when the weight cost $w_{i,j}$ is close to 1, the points $\mathbf{x}_i$ and $\mathbf{x}_j$ are likely to belong to separate regions, as judged purely from local evidence.

The weighting function $w$ is estimated comparing feature value at a point $\mathbf{x}_i$ with its neighbors. For features that are invariant under rigid transformation, we use the Euclidean distance between points in feature space as the edge weights. In this case, we consider only the case of one dimensional feature space, since multi dimensional feature spaces have usually different abstraction levels and types, their vicinities are relative to different scales of interests, and the use of mixte distance results in bias problem. Example of possible features are local curvature [HJBJ+96, ULVH06a, FJ89] and normal cones [ADC07a]. These features measure how gently or strongly curved a surface is at point, and, as outlined in the previous chapter, the quality of these estimates depend on the accuracy of the input data as well as the method used to compute them.

When the feature is the normal vector, which depends on the reference frame, we define the dissimilarity function as the angular difference in the case of normal orientation, $w_{i,j} = \angle(\mathbf{x}_i, \mathbf{x}_j)$. As showed in the previous Chapter, the normal estimation is quite accurate even when the data is corrupted with noise, which is the case of raw point clouds. Thus, this similarity function provides a reliable measure of the local surface curvature, and it is used in most of the experiments showed in this Chapter.

To illustrate the influence of the feature space and the dissimilarity function on the segmentation, in Figure 3.10 we show two segmentations when the similarity function is computed in different feature spaces. The similarity functions were the angle between the normals (Figure 3.10 (a)), when the cost function is set as $w_{i,j} = \angle(\mathbf{x}_i, \mathbf{x}_j)$, and the Euclidean distance in Gaussian curvature space (Figure 3.10 (c)), when the weight function is set as $w_{i,j} = \|K(\mathbf{x}_i) - K(\mathbf{x}_j)\|$. Observe that, even though both feature spaces consist of local shape descriptors, the segmentation does not produce the same regions. This is because the information embedded in each feature space are not equivalent. While differences between normals is proportional to local curvature amplitude, Gaussian curvature provides convexity/concavity information.

In the case where any other feature space is more relevant and more discriminant than local shape feature spaces, it is preferred in the mapping function. For example, some acquisition systems provide color or/and texture information, and for sampled surfaces with poor geometric information, or high symmetry, such color spaces are preferable. Other possibility, frequently used in segmentation community, is to approximate the similarity function as an approximation of a statistical distribution and the use mixed similarity functions [JBM+00, BFL06, YFM01].

In fact, region homogeneity predicate relies on the assumption that the *path* cost function is monotonic-incremental, which satisfy:

$$s(\tau \cdot (\mathbf{x}_i, \mathbf{x}_j)) \geq s(\tau) \tag{3.18}$$

$$s(\tau') \geq s(\tau) \Rightarrow s(\tau' \cdot (\mathbf{x}_i, \mathbf{x}_j)) \geq s(\tau \cdot (\mathbf{x}_i, \mathbf{x}_j)) \tag{3.19}$$

$$\tag{3.20}$$

where $\tau$ is a *path*, $s(\cdot)$ is the *path* cost function, and $\tau \cdot (\mathbf{x}_i, \mathbf{x}_j)$ is the concatenation of two *paths*. This property guarantees that, by removing an edge $(\mathbf{x}_i, \mathbf{x}_j) \in \Gamma_G$, the cost of the resulting

$\Gamma_G$ is reduced. Such property is the basic assumption of $MST$-based segmentation, where region homogeneity is achieved by removing edges in $\Gamma_G$ that connect different isolevels in $I$. A counterexample of weight cost function that is not monotonic incremental, which causes the segmentation to fail, is when the edges weight are allowed to be negative.

### Segmentation properties

We now turn to the complete raw point cloud segmentation algorithm. We resume the main properties of the algorithm and we want to show that, although the algorithm makes only greedy decisions, it produces a segmentation that satisfies global properties.

**Region homogeneity.** The $MST$-based segmentation algorithm produces regions that extremizes the criterion function (3.10). The criterion function, defined by the *Maximal internal difference*, $Dif_{\max}(R_i)$ (3.10), is a smoothing parameter that specifies the minimum radius of dilatation, in the feature space, necessary to connect points into the same region. Similarly, the homogeneity predicate, $\bar{P}(R_i, R_j)$ (3.11), specifies the minimum radius of dilatation necessary to connect, at least, one point from $R_i$ to one point from $R_j$. This interpretation is close related to the mean shift method [CM02], which is a non-parametric "feature space analysis" segmentation algorithm. In [CM02], a similar dilatation approach is used to group points in feature space, where points are grouped if they are up to a kernel radius apart from the same mode of the probability density function (p.d.f.) in feature space. The difference between these approaches is that our approach is less sensitive to input parameters than the mean shift approach [UPH07].

Finally, the regions are represented by a set of *paths* aligned to the isolevel of $I$, where the homogeneity predicate (3.15) is verified over this path. For small neighborhood size, these regions represent a compact surface in $\Sigma$.

**Input Parameters.** There are three input parameters in our segmentation algorithm: the neighborhood size, $r$, the minimal partition cost, $\epsilon_{\min}$, and the minimal region size. The neighborhood size, or the sphere radius $r$, is taken proportional to the average sampling spacing. The minimal region size is taken as a constant proportional to the point cloud size, which varies in the range $[0, 100\%]$. The threshold is taken as the value on the p.d.f. function that guarantees to cover % of the input edge set, and it varies in the range $[0, 100\%]$.

**Number of regions.** The $MST$-based segmentation algorithm produces an undetermined number of regions. The parameter $\epsilon_{\min}$ is the factor that influences the quantity of regions, and consequently, the level of details extracted.

**Robustness to noise.** Although the segmentation algorithm makes greedy decisions and the dataset partition is done according to local feature analysis, the influence of noise in the $MST$ configuration is well established, as presented in section 3.5.2. We assume that the dataset is dense and the influence zone of noise is limited to a small area and it does not change the global topology of the $MST$, thus preserving the boundary edges. Since noisy points are placed on

leaf nodes in the $MST$ by setting an appropriate value for the minimal region size, their effect on the final segmentation is minimized.

**Scalability.** The memory cost of our $MST$-based segmentation algorithm consists basically on the *neighborhood graph*, $G_{\mathcal{N}}$, construction, that requires $O(mn)$, where $m$ is the number of edges and $n$ the dataset size. Since the segmentation consists in recursively traversing the $MST$, the running time of the segmentation algorithm can be done in $O(n \log n)$ in an efficient implementation [FH04].

## 3.6 Evaluation of the algorithm

(a) (b) (c) (d)

**Figure 3.11:** *Too fine, too coarse and neither too fine nor too coarse segmentation. Given a point cloud (a), a data-driven segmentation can be classified as: (b) neither too fine nor too coarse, (c) too fine and (d) too coarse.*

We demonstrate our $MST$-based segmentation algorithm on several generic examples. The main goal is to prove the correctness and the robustness of the segmentation algorithm when partitioning *scans* raw point clouds. We show the original data and segments generated by our technique for a given set of input parameters. We present segmentation results in different formats, depending on what is more appropriate in each case.

Before we show the experimental results, we will define the criteria used to evaluate a segmentation. Since there is no assumption about the surface models composing the *scene*, it is impossible to determine a consensual segmentation ground truth. The segmentation rarely corresponds perfectly to objects. These relationships between segmentation regions and objects can vary with the algorithm's parameters, with the point cloud used, in fact they can even vary within a point cloud. As a consequence, the segmentation produced by data-driven methods cannot be evaluated as the model-driven approaches. The segmentation is classified according to the level of details associated with regions, and they fall into three categories [FH04]: neither too fine nor too coarse, too fine and too coarse segmentation. A segmentation is too fine (Figure 3.11 (c)) if there is some pair of regions $R_1, R_2 \in \mathcal{P}$ for which there is no evidence for a boundary between them. Oppositely, a segmentation is too coarse (Figure 3.11 (d)) if there exists a proper refinement of $\mathcal{P}$ that is not too fine.

Figure 3.12 shows the results of our segmentation algorithm on various point clouds. The


**Figure 3.12:** *Segmentation results for a variety of point clouds.*

results show the main properties of the $MST$-based segmentation algorithm, which is to generate regions of arbitrary shapes and to partition under boundary evidence. The feature space is the normal vector and the criterion function to be optimized is the angle between normals. The *scenes* represent a variety of surface types, from CAD geometric models, to free-form, man-made objects and roughness surfaces. Note that, in the case of the synthetic data, composed only by planes, the $MST$-based data-driven segmentation algorithm agrees with the segmentation ground truth. For the other *scenes*, they were taken by different acquisition systems, with different sampling density and noise model, and we verify the property of the algorithm to partition regions that represent compact surfaces, the roughness surfaces do not cause too fine segmentation, neither ill-defined boundaries causes too coarse segmentation.

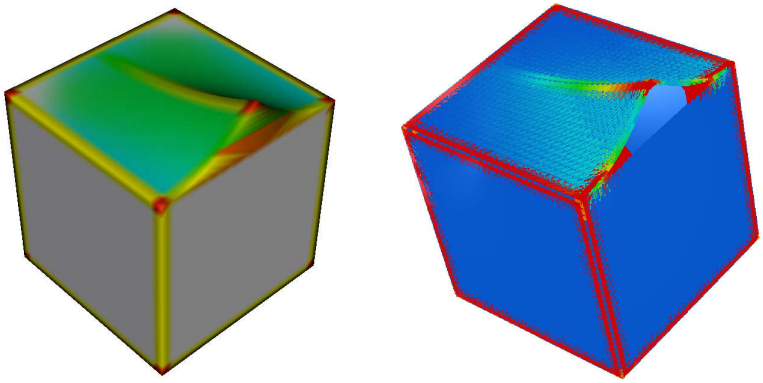
### 3.6.1 Performance evaluation

To evaluate the $MST$-based segmentation algorithm for the characteristics proposed in section 3.2, we perform a set of experiments on both synthetic and *scans*. To measure the correctness of the algorithm, we will generate multiple segmentations of each point cloud with multiple parameter settings. For each point cloud, the segmentation that best corresponds to a neither too fine nor too coarse segmentation is an approximation of the best performance possible.

The first experiment examines the parameter stability of the segmentations produced by the the $MST$-based algorithm with all possible input parameters configuration. Since we are only

interested in segmentations that produce regions representing compact surfaces in $\Sigma$, we set the neighborhood size fixed as three times the average sampling spacing.

Figures 3.13 and 3.14 show the results of our $MST$-based segmentation algorithm on a synthetic point cloud. In Figure 3.13 (a) is showed the curvature colormap, and Figure 3.13 (b) is showed the edges weight colormap of the *neighborhood graph*. In Figures 3.14 (a), (b), (c), are illustrated three segmentations using different parameters. Looking at the graph in Figure 3.14(d), that shows the resulting number of regions for a set of all possible input parameters, we can observe the interval of the parameter where the segmentation is neither too fine nor too coarse is quite large for this model.



(a) Curvature colormap (b) Graph edges weight colormap

**Figure 3.13:** *Synthetic dataset used to evaluate parameter stability.(a) Curvature colormap and (b) graph edges weight colormap.*

In Figure 3.15 we show the results when the same input parameters are used to segment three raw point clouds. The Bali *scans* are from our database, and they were acquired using the Breuckman TRITOS HE 100 (60x60x3 $\mu m$ resolution). They are noisy point clouds, large dataset (about 300.000 point per *scan*), and with color information. The region homogeneity predicate was set as a hard threshold, and the partition was performed when the edge weight in the $MST$ was above 5 degrees. We did not constraint the minimal regions size. It is observed missing data, holes, noise and outliers in every *scan*. Looking at the segmented regions obtained on all three segmentation and comparing them, we can notice that the algorithm is repeatable, once it generates similar regions, with similar feature variation, for equivalent surfaces. The differences observed in the obtained regions were mainly due to missing data. We can also notice that details in different scales were correctly extracted by our algorithm. Some small and meaningless surfaces are generated especially because of boarder points. These points have generally smaller density compared to the rest of the point cloud, and consequently, local shape descriptor estimation is less accurate over the entire region, and it is not recognized by our algorithm as noise.

### 3.6.2 Comparison with related graph-based segmentation methods

A comprehensive comparison of our algorithm with all the existing state-of-the-art approaches is beyond the scope of this thesis. From the 3D segmentation literature, we selected the Ncut

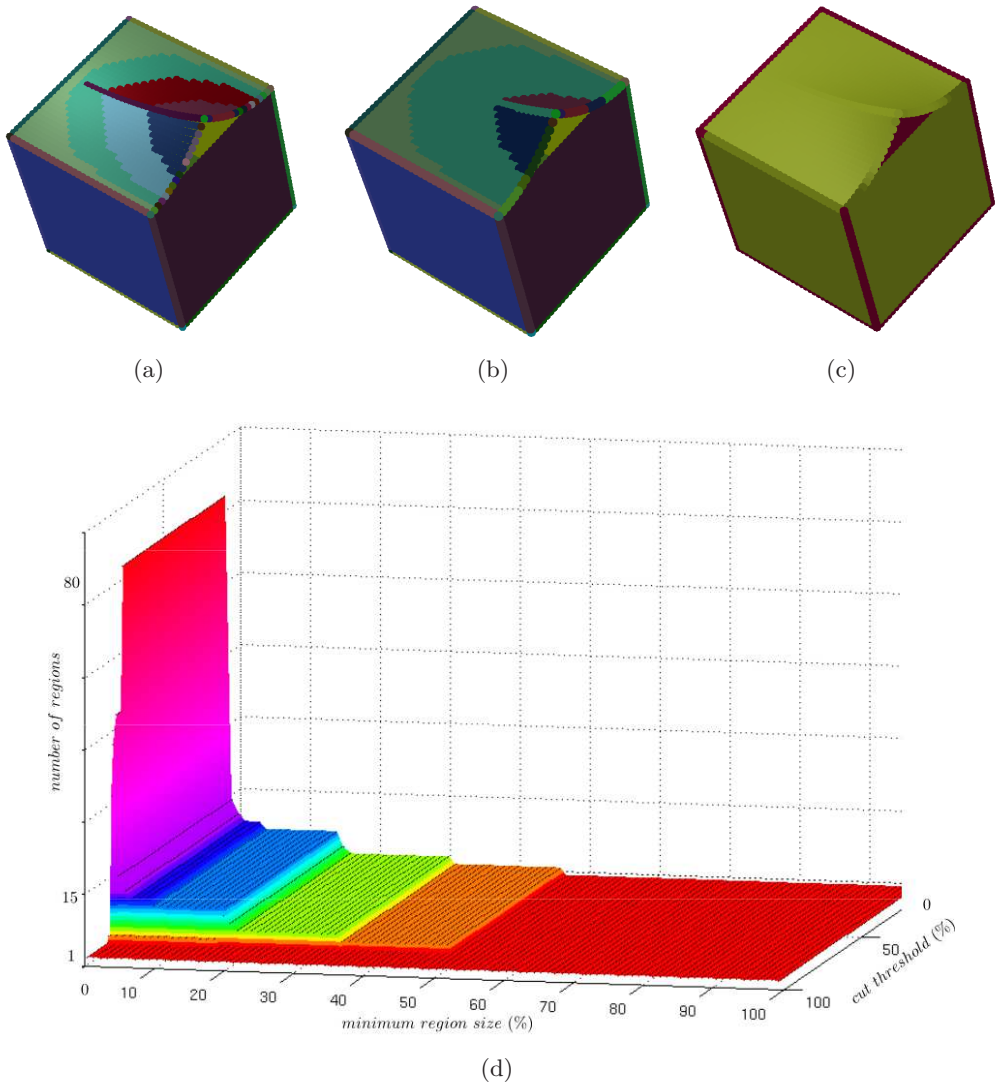



**Figure 3.14:** *Influence of the parameters on the segmentation results. We show the parameters intervals where the algorithm produces too coarse, too fine, and neither too coarse nor too fine segmentations.*

algorithm for evaluation purposes, primarily because it is also a graph-based approach and, contrarily to our approach, they guarantee to converge towards the global minima of the criterion function. We use the original *Ncut* segmentation algorithm designed to 2D image segmentation [SM00], instead of its extension to 3D point cloud segmentation [YFM01]. This choice was motivated by the fact we aim at comparing the performance of the algorithms without any pre or post processing. Besides, we want to take the same input graph in both segmentation algorithms, and thus the same search space of possible partitions. The drawback of using the original *Ncut* is that it limits the segmentation to only small models (up to 10.000 points).

The input parameters for the *MST*-based segmentation algorithm are set $\epsilon_{\min} = 40\%$ and minimal region size is 1% of the point cloud size. The input parameter of the Ncut segmentation algorithm is the number of regions. We set this parameter as the number of regions generated by our *MST*-based segmentation algorithm.

In Figure 3.16 is showed some segmentation results for two synthetic point clouds. These

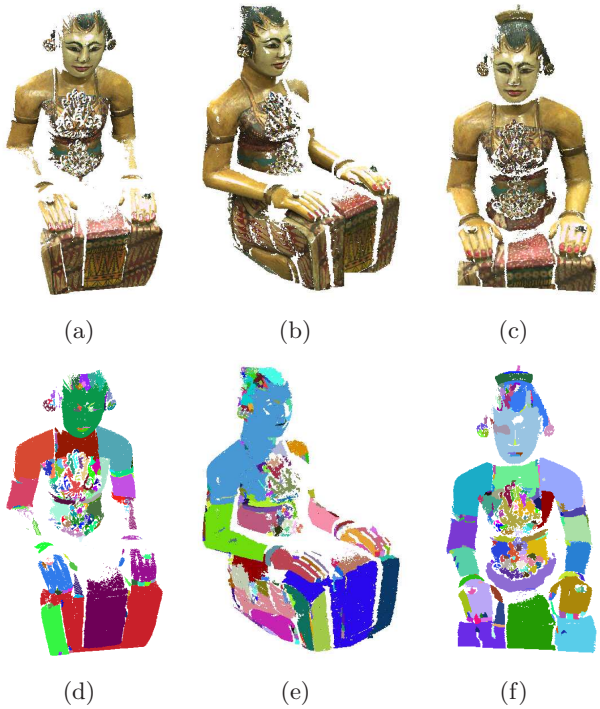
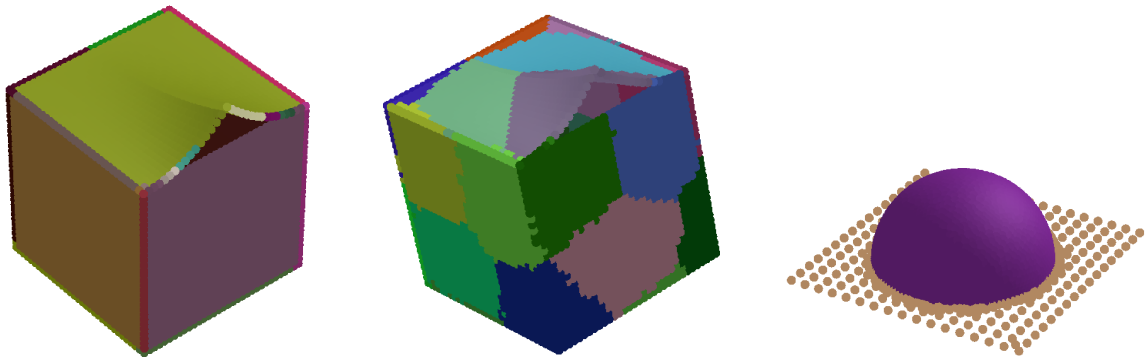



**Figure 3.15:** *Parameter stability and scalability evaluation. (a), (b), (c) Original point clouds and (d), (e), (f) their respective segmentations. The feature space considered was the angle between normals and a hard threshold is taken as the partition criterion.*

results clearly outline the properties of both graph-based segmentation algorithms, their similarities and their differences. In Figure 3.16 (a), (b), the segmentation are quite different from each other. While the $MST$-based segmentation algorithm tries to find a partition of a weighted graph under boundary evidence, the $Ncut$ approach aims at finding the optimal partition that produces regions of similar sizes. In Figure 3.16 (b) we observe the main drawback of the Ncut segmentation algorithm: the optimal *cuts* not always follow the surface boundary, since the noiseless plane is partition into four regions and the high small feature on the top face is not break into two, as expected.The segmentation results in Figure 3.16 (c), (d) agree, since the point cloud is partition into two regions of balanced sizes.

In Figure 3.17 we show some visual comparison of segmentation results obtained with the methods $Ncut$, two watershed variants and ours. The Stanford Bunny dataset is a complex model, because of the presence of roughness regions and the smooth boundaries that can induce an ambiguity in boundary evidence evaluation and, consequently, the "leaking" effect. Re-sampling reduces the size of the Stanford Bunny dataset to 7.000 points. The normals are estimated from the triangle mesh representation. The input parameters for the $MST$-based segmentation algorithm are $\epsilon_{\text{min}} = 40\%$ and minimal region size is 1% of the point cloud size. Although our $MST$-based segmentation algorithm converges towards the local minimum, it extracted meaningful regions that satisfy global properties (Figure 3.17 (a)). However, because of the surface roughness, the region boundaries of our segmentation have irregular shapes, and we can observe the "leaking" effect on the right ear. Again, the Ncut segmentation algorithm produced regions that do not follow the shape natural boundaries, like the tail, and the segmenta-



(a) $MST$-based: 21 regions (b) Ncut: 21 regions (c) $MST$-based and Ncut: 2 regions

**Figure 3.16:** *Comparison between MST-based and Ncut segmentation algorithms on a synthetic noiseless dataset. (a), (c) The MST-based segmentation and (b),(d) the normalized cut segmentation results.*

tion extracted only the global shape structure (Figure (b)). Both watershed variants performed poorly on this dataset, mainly because of their inability to handle "textured" surfaces (Figures 3.17 (c),(d)). The results of watershed algorithm were taken from two variants [PKA03, JM07], and there is no control about their input parameters.

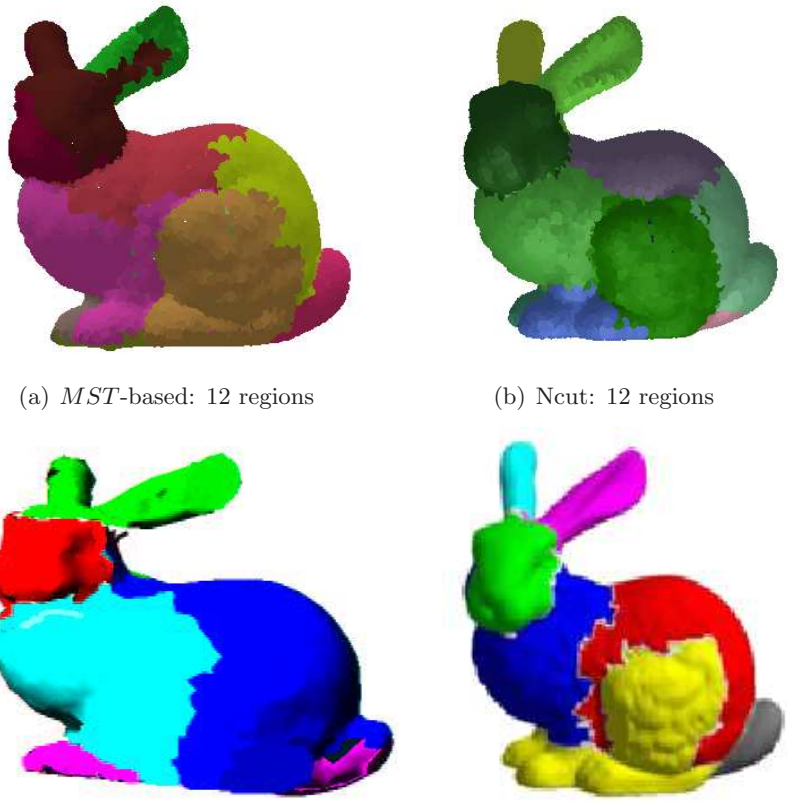
Figure 3.18 shows the segmentation using the $MST$-based and the Ncut algorithms when the input data is heavily corrupted with a zero mean Gaussian noise. We can observe that both algorithm performed poorly. The noise affects not only the *neighborhood graph* connectivity, but also the feature estimation. This effect is amplified by the fact the data is sparse. The instability of the $MST$-based segmentation is observed mostly on the "texture" surface, where the boundary is hardly detected, and more regions are generated for the noisy model, from 12 to 14 regions. However, the Ncut algorithm, even if the number of regions remains the same, produces a segmentation far from the noiseless case.

### 3.6.3 Application of the algorithm to other datasets

The $MST$-based segmentation algorithm can be applied to any $N$-dimensional image. Volumetric images, for example, are commonly obtained from devices such as CTs and MRI. It is composed by a set of 2D images, grouped in layers in order to form the volume. Point coordinate $\mathbf{p}_i = (x, y, z)^T$ is directly obtained by image pixel coordinate and the relation between successive image layers. Gray level embeds material properties. Segmentation breaks the input image into volumes, where each region has an homogeneous material property.

If we consider time as a third dimension in our neighborhood graph, the method can be extended to video sequence segmentation. The initial dataset is indeed composed of a video stack. The problem with this approach is the sampling frequency inconsistency between spacial and temporal dimensions. To overcome this difficulty, we set the time step to one. This extension can only be envisaged when dealing with offline video segmentation because all the pixels must be available for the neighborhood graph construction.

Figure 3.19 shows the original and segmented frames of a video sequence [Ohi]. Note that the texture on the background did not caused too fine segmentation. This kind variability is



(a) *MST*-based: 12 regions (b) Ncut: 12 regions

(c) Watershed variation [JM07] (d) Watershed variation [PKA03]

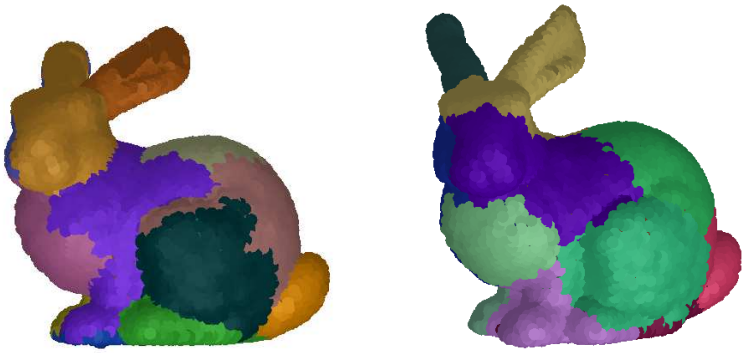
**Figure 3.17:** *Comparison between the MST-based, the Ncut and the watershed segmentation algorithms. Segmentation results when segmentation is performed using the (a) MST-based, (b) the normalized cut, and two variations of the watershed segmentation algorithm (c), (d).*

handled by choosing a large neighborhood size and the adaptive threshold.

## 3.7 Conclusions

In this chapter we have proposed an algorithm for raw point cloud segmentation. We have established a set of requirements necessary to any segmentation algorithm in order to produce useful and predictable results. We consider five main properties: correctness, stability with respect to parameter choice, robustness to noise, scalability and data-driven. We propose a taxonomy for 3D segmentation algorithms, taking into account both criterion function to be optimized and the technique used. Among all segmentation algorithms, we concentrate ourselves to data-driven, graph-based segmentation algorithms, since this is the category in the taxonomy that best fulfills the segmentation requirements to raw point cloud segmentation.

We then extended a graph-based segmentation approach, originally designed to 2D image segmentation [FH04], to work on raw point cloud. We evaluate the performance of the algorithm on both synthetic and real raw point clouds. The underlying numerical optimization scheme is straightforward and robust. It is easy to implement, not expensive to compute, robust across many noise levels and parameter settings, and useful for segmentation of raw point clouds. From



(a) $MST$-based: 14 regions (b) Ncut: 12 regions

**Figure 3.18:** *Comparing the robustness of (a) our MST-based segmentation and (b) Ncut when the data is corrupted with noise.*

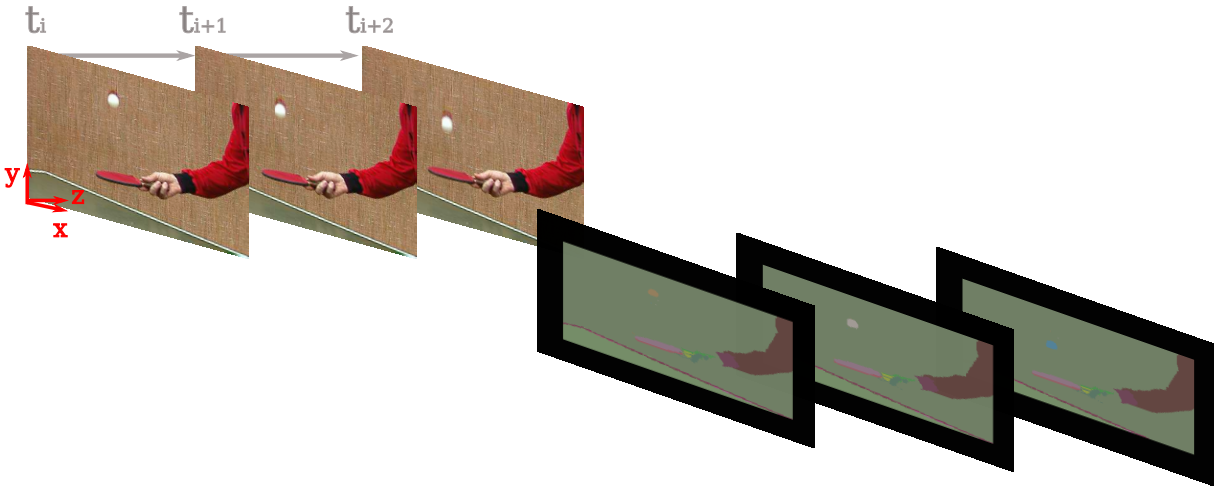



**Figure 3.19:** *Video Segmentation. (a) Original range image and (b) segmented image in RGB color space.*

our experiments regarding the performance of the $MST$-based segmentation algorithm in terms of parameter stability and correctness of its results, we conclude that the algorithm produces correct segmentation for a quite large parameter interval, and, given a stable parameters set, it provides acceptable segmentation for different input shapes and noise levels. The segmentation results on raw point clouds *scans* were repeatable, even in the presence of missing data and noise.

Another contribution was the comparison between our $MST$-segmentation with one of the standard graph-based segmentation algorithms, the $Ncut$. We verified that the $Ncut$ is incapable of extracting regions of general sizes and it does not partition following the surface boundaries. Besides, it requires the number of regions to be set *a priori* and it does not work on large datasets. These drawbacks were not verified when segmentation was performed using our $MST$-based segmentation algorithm. These discoveries motivate our approach to using segmentation in a global registration scheme, which will we presented in the next Chapter.

# Chapter 4

# Global pairwise registration based on region correspondence

## 4.1 Introduction

In this chapter we investigate the problem of pairwise global registration. As discussed in Chapter 1, paiwise global registration is the problem of optimal alignment of two *scans* in arbitrary initial positions and it is the first stage towards obtaining a 3D digital model from a *scan* set. We are interested, in particular, in registering raw noisy data, possibly contaminated with outliers, without pre-filtering or denoising the data.

We introduce a global registration algorithm [ADC09] based on region correspondence, an efficient scheme for raw point clouds, which is resilient to noise and outliers. We will incorporate the regions obtained from the segmentation of *scans*, using the method presented in Chapter 3, into this higher level application. The originality of our work is to use region as data representation. Instead of using salient features to find correspondence between *scans*, we want to incorporate all features available. This approach aims at solving the problem of comparing inconsistent, noisy features, or comparing only features that are not in the overlapping area. Thus, region representation is preferred because of its descriptive power and because it also reduces the data volume treated and inimizes the noise effect on the pipeline.

Before we present our global registration algorithm, we will pose the registration problem (Section 4.2). We will examine each one of the sub problems envolving the global registration of two *scans*, and which strategies where proposed in the literature to solve these problems. We outline their advantages and their drawbacks. Then, we will present the formalization of our global registration algorithm (Section 4.3), the hypothesis made about the data, about the regions representing the data, and the analysis of the performance of the algorithm.

## 4.2 Foundation of pairwise global registration

The global registration, or *scans* alignment, is the process of matching the common overlapping area between two 3D *scans* in arbitrary initial position and estimating the corresponding alignment. This is an important problem not only to 3D model acquisition, but also to 3D object recognition, geometric processing, 3D object indexing, motion capture, shape morphing, texture transfer, and statistical shape analysis. In this section, we present the basic concepts, hypothesis and main works done on pairwise global registration.

### 4.2.1 Registration problem

Let $S_A$ and $S_B$ be a *scan* pair, consisting of $\{\mathbf{p}_1, \cdots, \mathbf{p}_N\}$ and $\{\mathbf{q}_1 \cdots \mathbf{q}_M\}$ points, respectively. We call $S_A$ the *source* and $S_B$ the *target scan*. We assume that, at least partially, the surfaces represented by the point clouds overlap. The goal is to find the *rigid transformation* that best registers the *source* with respect to the *target*.

Let $\{(\mathbf{p}_i, \mathbf{q}_i)\} \subset S_A \times S_B$ be a set of corresponding points. The best rigid transformation that brings both point clouds to the same reference frame is the one that minimizes the *coordinate root mean squared error* ($cRMS$):

$$cRMS^2(S_A, S_B) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|^2 \tag{4.1}$$

where the rigid transformation is represented by the translation vector $\mathbf{t}$ and the rotation matrix $\mathbf{R}$, and $\|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|^2$ is the distance from the transformed *source* point $\mathbf{p}_i$ to its corresponding point $\mathbf{q}_i$ in the *target scan*.

Theoretically, if we can correctly establish correspondence between just three points in a non degenerate configuration across the *target* and *source*, then we can uniquely solve the aligning rigid transformation. A brute-force approach to testing all possible alignments involves selecting a pair of triplet points, $\{(\mathbf{p}_1, \mathbf{q}_1), (\mathbf{p}_2, \mathbf{q}_2), (\mathbf{p}_3, \mathbf{q}_3)\}$, computing an initial alignment based on these points, to recover a candidate matching error $cRMS$, and repeating the process for all pairs of triplets points distributed around each *scan*. This naive alignment scheme has a time complexity of $O(M^3N^3)$, and it requires to compute the transformation for the entire set [GMGP05],which becomes overly expensive as the number of regions grows.

The main idea behind global registration algorithms is to reduce considerably the set of possible corresponding pairs $(\mathbf{p}_i, \mathbf{q}_i)$ and explore it efficiently in order to find a subset $\{(\mathbf{p}_i, \mathbf{q}_i)\}$ that minimizes the alignment error $cRMS$ defined by (4.1). In the next section, we present some of the proposed solutions to the global registration problem and we examine their strategies to reduce the solution space and to be robust to noise and partial overlapping.

### 4.2.2 Pairwise global registration: a review

There is an abundant literature for the problem of automatic pairwise global registration [PMW05, HP05, HFG+06, GMGP05, HCH00, MGGP06, Mit06]. The general pipeline of global registration algorithms is illustrated in Figure 4.1.

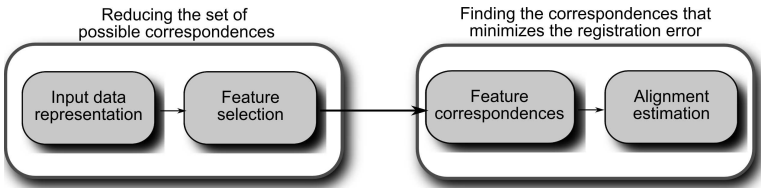



**Figure 4.1:** *Global pairwise registration pipeline. First, data are represented in some feature space, then discriminant features in both scans are independently identified, and feature correspondences is established in order to compute the rigid alignment transformation.*

The first part of the pipeline aims at reducing the search space of possible correspondences. To this end, initially, it is derived a concise representation for the input data. It can be geometric features associated with points [PMW05], or descriptors associated with subsets of the input data [HP05, HFG$^+$06]. These features must be invariant to translation, rotation and robust to re-sampling. In a typical *scan*, the majority of the features are redundant and, thus, an intermediary step, called feature extraction, identifies the discriminant features in both *source* and *target*, independently. In a feature extraction algorithm, discrimi nant features are those whose values are rare among the input data. To prevent noise, discriminant features are the ones that remain rare even when computed in different scales [GMGP05]. Represent the input data as a subsample not only reduces the set of potential correspondences, but it also removes featureless descriptors which can induce the algorithm to converge towards a local minimum.

The second part of the pipeline aims at efficiently exploring the search space in order to find the correspondence set that minimizes the registration error. After feature extraction, features from different *scans* are matched according to their similarity in feature space. It gives the initial set of potential correspondences. Even in the case that local shape descriptors are perfectly distinctive, symmetries in the models can give rise to multiple correspondences, and one-to-one correspondence based on the comparison of feature descriptor values is rarely found. Additional hypothesis about inter features relationship, or geometric consistency, must be verified to reinforce the potential set of correspondences. This additional constraint prunes some wrong correspondences and only a small subset of potential feature correspondences are left. At this point, the set of potential correspondences is considerably reduced, and we can afford to compute the aligning transformation for each correspondence subset, with the associated registration error $cRMS$. Given the subsets of potential feature correspondences, the rigid transformation computed to align these features and their associated $cRMS$, the initial alignment is chosen to be the rigid transformation with the smallest error $cRMS$ associated.

Global registration algorithms differentiate from each other by the techniques used and the assumption made at each stage of the pipeline. If we take into account only the first part of the global registration pipeline, any algorithm falls into one of two general classes: feature-based and region-based approaches. If, however, we take into account the second part of the global registration pipeline, the algorithms are characterized by the corresponcence techniques they use. Examples of correspondence search algorithms are graph optimization [SOS05, HH01], foward search [HFG$^+$06] and branch-and-bound-based combinatorial search [GMGP05]. We will focus on the first stage of the pipeline, because this stage is crucial to the correctness and the

robustness of the algorithm.

A summary of the most popular global registration algorithms is showed in table 4.1. The first observation is that most of the past research efforts concentrated on finding local shape features correspondences to estimate the initial alignment [PMW05]. Region-based matching has gained much less attention, partly because of the lack of segmentation algorithms which produce repeatable regions [VH08]. While the first approach focuses mainly on the choice of a robust surface descriptor, on selecting rare local surfaces descriptors and comparing them over different scans, region approaches are less depend on the local feature itself and more dependent on the feature distribution over the region. We will examine some of the works done in each category, in the context of large raw point cloud registration.

### Feature-based global registration algorithms

Feature-based approaches are categorized according to the dimension and the level of detail of their local shape descriptors. The performance of such techniques is directly related to the feature estimation process, to the robustness and to the repeatability of this estimation under noise and re-sampling. The correctness of feature-based registration algorithms also depend in the strategy used to extract discriminant feature, which must guarantee that some of features used to find correspondence are in the overlapping area. A survey on *scans* pairwise initial alignment based on features correspondence is presented in [PMW05].

Johnson in [JH97], introduced the spin image, a high-dimensional local geometric descriptor. The spin image is a 2D histogram of surface locations around a point, which provides a fairly detailed description of the shape around the point. It is the most popular high-dimensional feature and was used to other applications such as 3D recognition [FHK$^+$04], shape similarity and retrieval [MGGP06]. The original algorithm to compute this feature was later modified by Huber in [HCH00] to treat large and non-uniform datasets. However, neither of the proposed spin image variants are sufficiently robust for partial surface alignment. Compared to other global registration algorithms, the spin image matching has the advantage that, given a discriminant point in one *scan*, it is likely that it will be found only a few points with similar descriptors on the second *scan*.

Low-dimensional descriptors, on the other hand, compute only a few values per point. Examples of such descriptors include curvature-based quantities [HJBJ$^+$96, ULVH06a, FJ89], shape index [DJ97], integral descriptor [GMGP05, HFG$^+$06]. Since they are not discriminant, matching low-dimensional descriptors doesn't guarantee one-to-one robust feature correspondence. To solve this problem, the global registration methods using these descriptors performed robust feature extraction, in order to find rare features used in matching process. From all low-dimensional descriptors, the integral invariant [PWHY09] has been largelly used in global registration algorithms [GMGP05, HFG$^+$06]. This local shape descriptor provides curvature information, and its estimation is based on performing integral operations on the underlying shape. It was proved the robustness of this shape descriptor for uniform smooth meshes, but a robust estimation for non-uniform data and corrupted with noise is still an open problem. Another drawback of using features to compute correspondences is that global registration algorithm using this data rep-

| Algorithm | Input dataset | Point Feature | Data representation | Matching function | Correspondence search | Large Dataset |
|---|---|---|---|---|---|---|
| Johnson in [JH97] | Uniform sampling mesh | Spin image | Discriminant features in model and entire dataset on data | Spin image correlation | No | Discriminant spin image in *target* and entire dataset on *source* |
| Gelfand in [GMGP05] | Uniform sampling mesh | Integral invariants | Rare features in both data and model | Euclidean feature distance | Branch-and-bound search | Rare features in both data and model |
| Huang in [HP05] | Uniform sampling point cloud | Integral invariants | Homogeneous regions | Shape descriptor Euclidean distance | Foward search | Entire region set |
| Our method | Non-uniform point cloud | Local descriptor | Non-homogeneous regions | Region histogram comparison | Branch-and-bound search | Entire region set |

**Table 4.1:** *Summary of some global registration algorithms.*

resentation suffer from the problem of picking inconsistent features on the input dataset, since the two datasets are processed separately.

#### Region-based global registration algorithms

Region-based global registration has resemblance to the works on partial shape matching [GCO06, OFCD01, TV04, TV08]. Compared to the shape matching problem, where the goal is to define a similarity function to compare shapes, the global registration problem has the additional registration issue. To our knowledge, only one region-based global registration algorithm has been previously proposed in the literature [HP05]. In fact, the use of region has not been very popular and it has even been avoided in tasks that require region matching. This is because of the difficulty found in describing robustly the region, in generating repeatable regions in different *scans*, and matching partial overlapping regions.

In [HP05], Huang and Pottmann proposed a complete pipeline to solve the registration problem. They partition the input dataset into homogeneous regions in feature space, using as feature the integral invariant [PWHY09]. The success of their region matching rely on the use topological relations of regions to prune incorrect correspondences, where regions hierarchy is used to select the set of region correspondences. They use regions to find a rough correspondence between regions, and these correspondences are later refined by a feature correspondence strategy, close to the one developed in [GMGP05]. Such an approach increases the complexity of the algorithm, but the use of region correspondences to guide feature extraction improves the robustness of the algorithm to partial matching. The main drawbacks of this global registration algorithm are that it is not efficient to large data, and there is a large number of tuning parameters. Besides, both segmentation and feature extraction are not robust to noise, since noise can drastically change regions configuration because of feature variation, which affects both region and feature correspondences search.

Comparing the methods showed in table 4.1, we observe that all algorithms propose strategies to process efficiently large point sets and to be robust to both noise and partial matching. However, they all suffer from the same drawback: they do not guarantee that the initial feature set contains features located in the overlapping area. In such scenarios, there are several advantages of using regions over interest points for global registration. Region-based approaches are less dependent on sample density, noise and holes presented on the input data. Besides, once region reduces naturally the data volume, we can use the entire solution space to find the best correspondence set. Motivated by these arguments, we develop in the next section a pairwise global registration based on region correspondence.

## 4.3 Global registration as region correspondence problem

We formulate global registration as the problem of finding a set of region correspondences, and use these correspondences to compute the initial alignment transformation. Our algorithm explores the fact that the aligning transform is low-dimensional and only a small set of corresponding region pairs is necessary to specify the rough alignment of the *scans*.

Let $\mathcal{P}_{S_A} = \{A_1 \dots A_n\}$ and $\mathcal{P}_{S_B} = \{B_1 \dots B_m\}$ be the partition of the *source*, $S_A$, and the *target*, $S_B$, *scans*, respectively. The correspondence between two regions $A_i$ and $B_j$ is then establish through the *binary equivalence relation*, enunciated as follows: two regions $A_i$ and $B_j$ are equivalent with respect to $\bar{E}$, or $A_i \sim_{\bar{E}} B_j$, if it is observed both regions similarity according to their descriptor values, and geometric relations between regions within the *scans*.

The first condition states that global, or absolule, properties of corresponding regions must agree. Two regions, $A_i$ and $B_j$, potentially represent the same surface in the *scene* if they have similar absolute region properties. Inversely, if there is a large disagreement between the absolute region properties, is likelly that they represent different surfaces in the *scene*. Examples of global properties of a region are its shape, its area and the feature distribution. In Figure 4.2 is illustrated how absolute region properties change between *scans* and how it is established correspondence between these regions, when considering only the first equivalence condition. In this example, it is taken two *scans* that partially overlap and they are represented by a region set, as showed in Figure 4.2 (a). The absolute property that characterizes a region in this example is its shape and, in Figure 4.2 (b), we show some regions in both *scans* that represent the same physical surface. Regions have the same index if they represent the same physical surface. Here, the shape of the region $a$ is similar in both *scans*, and the comparison between these regions is done by directly comparing their shape descriptors. A different situation is observed between the regions $b$ and $b'$, where the shape of the regions $b$ and $b'$ matches only partially. Consequently, the global properties of the shapes $b$ and $b'$ change drastically and the comparison of their shape descriptors might not provide a good match cost. A third situation that also occurs frequently is showed in the region $c$. In the ideal case, $c$ should be matched to both $c_1$, $c_2$, and $c_3$. However, the absolute properties of $c_1$, $c_2$, and $c_3$ differ significantly as compared to their correspondence $c$, and these three regions should be merged before comparing their shape descriptors with $c$. From the equivalence definition, only the regions $a$ are in correspondence, since they are the only in which shape did not change drastically between the *scans*, and we say that there is one-to-one region correspondence. In the present work, we will not adress the problem of partial region matching, or many-to-one region correspondence. This one-to-one correspondence hypothesis may seem too restrictive, and we are aware that partial matching occurs frequently on real datasets, especially on noisy datasets and where missing data and holes occur. Since we aim at finding a minimum number of region correspondences necessary to compute the initial alignment, a partial region matching seems overly expensive [BAAC08, VH08], and unecessary for most of point clouds treated.

The second condition states that the geometric constraints given by the hypotheses made about the data and the transformation class must be verified between correspondences. Such condition is necessary to renforce the correspondences and filter wrong ones. Assuming that the *scans* represent only rigid objects, we have that the distance between any two points must not change. Besides, if only affine trasnformation is applied to the *scans*, we have that any rigid transformation has to preserve inter-point distance, which is based on the distance between two points on the same point cloud. Let $(A_1, B_1)$, $(A_2, B_2)$ be a pair of consistent region correspondences. From the rigidity hypotheses, we have that, the relative pose between $A_1$


(a) (b)

**Figure 4.2:** *Equivalence condition. (a) Two scans represented by a region set. (b) Correspondence between regions when the region absolute property is taken as its shape, where we outline some regions for better illustration of the concept. Matched regions have the same region index.*

and $A_2$ should be consistent with the relative pose between $B_1$ and $B_2$, even when they suffer some rigid transformation. If this geometric consistency is not verified for a pair of region correspondences, one of the correspondences, or both, are necessarily wrong. This last condition is an important supplementary constraint that avoids situations where symmetries and regions in the same sample with similar descriptors induce ambiguities on the correspondence process.

Based on these conditions, we formulate the global registration as the problem of finding a set of region correspondences that minimizes the registration error, where the corresponding regions verify the equivalence conditions. In the case of rigid motion, three corresponding points are sufficient to uniquely determine the alignment transform. Working in the region space, the alignment transform is determined by, at least, three corresponding regions. Thus, we developed an algorithm that searches for potential region triplets that are in correspondence and find the correspondence that minimizes the registration error. To be robust to noise and incorrect correspondences, other approaches enforces the matching by finding more than three correspondences. However, when working in region space, the number of correspondences required imposes the minimal size of the overlapping area. If we aim at finding more than three region correspondences, it implicitely requires a larger overlapping area. For this reason, we formulate the correspondence problem as the search for three region correspondences. The algorithm is illustrated in Figure 4.3. First, the raw point clouds are independently partition into regions, and each region has a feature associated to it. Then, we use the first equivalence condition to built the set of potential correspondences, which is done by matching the regions, in feature space. From the second equivalence condition, we prune the set of potential correspondences by verifying the geometric coherence between pairs of possible correspondences. We built triplets of corresponding regions that verify both equivalence conditions. We search efficiently the correspondences triplet which alignment transformation minimizes the registration error and this alignment is finally applied to the *scans*.

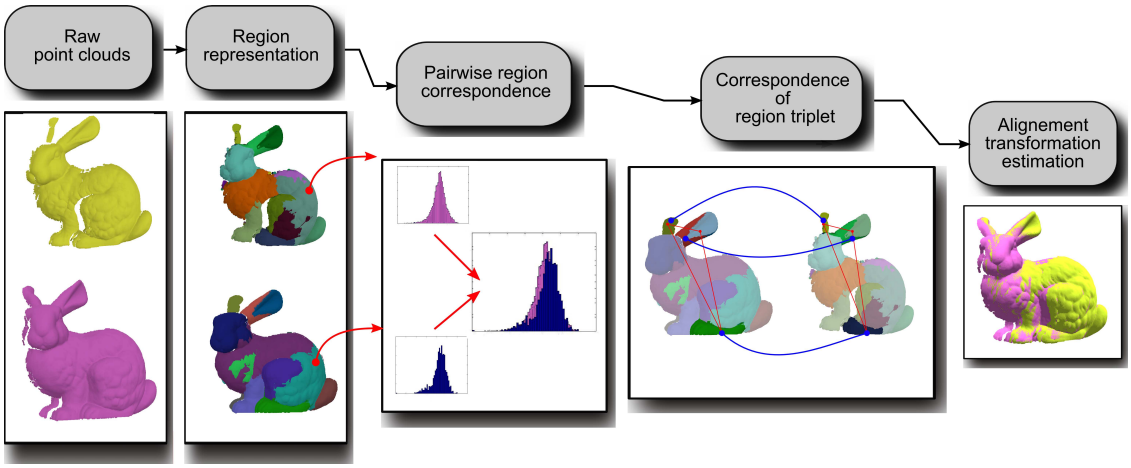



**Figure 4.3:** *Pipeline of our global registration based on region correspondence.*

### 4.3.1 Region representation

The computational complexity of establishing correspondences between models has motivated a large body of research in the area of shape and region descriptors [KFR03]. Two regions can be compared by independently computing their region representation and then defining the measure of similarity in terms of the distance between their descriptors.

To this purpose, we derive a concise representation of each region using a set of characteristic values, vectors, and points. A region is a subset of the point cloud, where points have both spatial coherence and some common property in the feature space. We store the following information for each region $R_i$:

$$R_i \longrightarrow \{\mathbf{c}(R_i), |R_i|, H(R_i)\} \tag{4.2}$$

- The point $\mathbf{c}(R_i)$ represents the *position* of the region, defined as the closest point to the barycenter of the region.
- $|R_i|$ is the size of the region $R_i$.
- $H(R_i)$ is the local shape descriptor histogram of points $\mathbf{x} \in R_i$.

The histogram $H(R_i)$ is a non-parametric representation of the probability density function of a region. It is a practical and reliable mean for approximation of the probability density function of the local descriptors and it contains the signature of the region $R_i$. This descriptor yields invariance under rigid motions and mirror imaging. In this case, invariance under scaling can be added by normalization of the histogram before comparing them and/or by factoring out scale during the comparison. Histograms have been used in both 3D global shape representation [AKpKS99, OFCD01] and high-dimensional local descriptors [JH97, ZIZ99]. We preferred a non-parametric p.d.f. representation over a parametric representation because non-parametric p.d.f. has the ability to discriminate, since it can model higher dimensional information and multivariate distributions.

Any point descriptor that is invariant under rigid transformation can be considered to compute $H(R_i)$. In our experiments, we use local shape descriptors, like Gaussian and mean curvatures and normal cone, but any other invariant feature associated with points can be used, such as shape distribution [OFCD01], color and texture. Once we have computed the local shape descriptor histogram for two regions, the dissimilarity between the regions can be evaluated using any metric that measures distance between distributions.

### 4.3.2 Region through segmentation

The global registration based on region correspondences implicitly rely on the regions generated by the segmentation to be equivalent among *scans*. It means that, given a *scan* set from the same *scene*, we say that the segmentation algorithm produces equivalent results if regions in the overlapping area conserve global shape properties among different *scans*. If this property is observed, we have that any object/part presented in different *scans* is partitioned into similar regions, even when local shape property varies among *scans*. In our experiments, we use the $MST$-based algorithm presented in Chapter 3. However, other data-driven algorithms can be used instead.

An example of correct segmentation, according to the above criteria, is showed in Figure 4.4, where the $MST$-based segmentation algorithm is used to partition a set of *scans* from the same object. All *scans* were segmented with the same input parameters. Note that there are some regions in the overlapping area where all segmentations are in agreement, and other regions where there are varying levels of disagreement. Segmentation differences are mainly due to missing data, holes or the presence of a few spurious points on the boundaries. In the presence of these artifacs, the original regions of the object leak out to form new regions and it is extremely unlikely to generate a segmentation of an entire *scan* that corresponds to human partitioning of the same *scene* [Pan08].

The main advantages of using the MST-based segmentation algorithm over other data-driven methods are that the solution space of all possible partitions is drastically reduced when using this approach and the fact that the noise can be detected and filtered during segmentation process [ADC07b]. Additionally, the MST-based segmentation algorithm bounds only local attribute variation and, consequently, the regions regenerated by this algorithm are not necessarily homogeneous in feature space. It allows more dynamic to the descriptors within a regions, which aids in regions differentiation.

### 4.3.3 Pairwise region matching

Having constructed the representation for two 3D regions, we are left with the task of comparing them. The region representation maps a complex region onto a feature vector in a multidimensional space. The similarity of two regions is then defined as the vicinity of their feature vectors in the feature space. In the literature, there is a large number of similarity functions to compare histograms, such as Euclidean distance, quadratic form distance [AKpKS99], statistical and probabilistic approaches [JH97, HLLS01], among others. The Euclidean distance exhibits severe limitations with respect to similarity measurement, since individual components of the

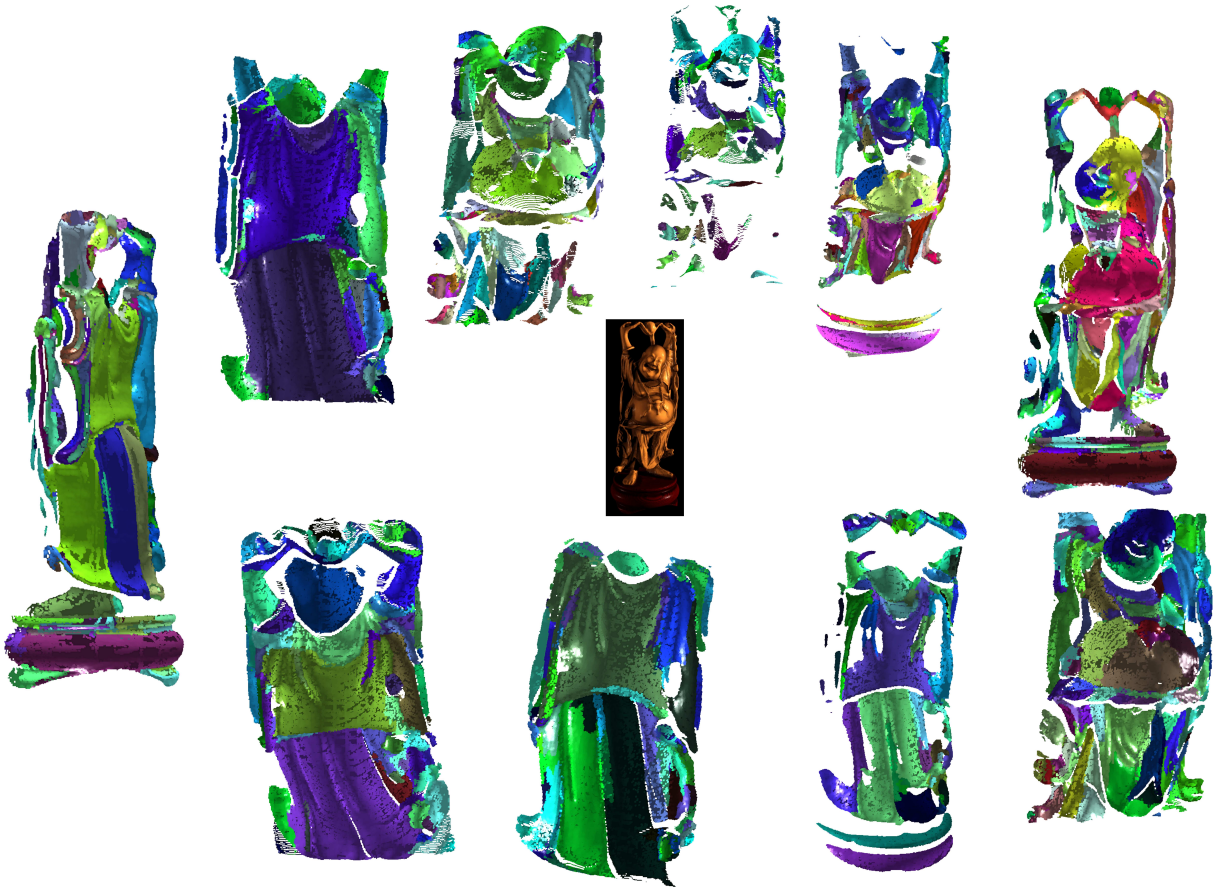


**Figure 4.4:** *Equivalence of point cloud segmentation. Given a set of scans, covering the entire object, the segmentation produce equivalent regions in the overlapping between scans.*

feature vectors are assumed to be independent from each other and a shift in the histogram is neglected by the Euclidean distance. In this thesis, we restrict ourselves to statistical similarity functions, where, given the distribution for a region in the *target*, the correspondence problem reduces to the task of finding the region in the *source* with the distribution that best match the region in the *target*. Compared to other similarity functions, statistical similarity functions is adapted to be computed and evaluated efficiently.

### Histogram matching

The formal statistical method for assessing the similarity between two probability is the $\chi^2$-test. We use an equivalent histogram distance function, the *intersection*-measurement between histograms, which is a measure that quantifies the common parts of two histograms. The intersection of two histograms $H(A_i)$ and $H(B_j)$ defines regions resemblance with respect to their corresponding signatures and it is given by [HLLS01]:

$$\cap\left(H(A_i), H(B_j)\right) = \frac{|A_i \cap B_j|}{|A_i \cup B_j|}, \tag{4.3}$$

where $|A_i \cap B_j|$ denotes

$$\frac{1}{\beta}\sum_k \min(a_k, b_k) \tag{4.4}$$

and $|A_i \cup B_j|$ denotes

$$\frac{1}{\beta}\sum_k \max(a_k, b_k) \tag{4.5}$$

where $\beta$ is the number of cells in the histograms and $a_k, b_k$ are the $k$-th histogram cell. Equation (4.3) assumes values in the range $[0, 1]$.

This measure does not take into account regions shape information or points location. This property is an advantage for matching partial overlapping regions, since small variations on shape and region size do not affect considerably the local shape descriptor histogram. Furthermore, each cell in the intersection distance function (4.4) is treated equally and are supposed to be equally probable. As a consequence, this distance function defavours regions of similar sizes.

From the histogram intersection function, we define the dissimilarity function between two regions $A_i$ and $B_j$ as:

$$Ds(A_i, B_j) = [1 - \cap(H(A_i), H(B_j))]^2 \tag{4.6}$$

We compute the dissimilarity function $Ds(A_i, B_j)$ for every pair $(A_i, B_j)$, with $A_i \in \mathcal{P}_{S_A}$, $B_j \in \mathcal{P}_{S_B}$ and the values are stored in a dissimilarity matrix. This dissimilarity measure is invariant under rigid transformation and robust to small perturbations. It also has the favorable property: the histograms can be subject to considerable noise before noise bias the region p.d.f. representation. On the other hand, if the region is not globally discriminative or unique, a matching algorithm based uniquely on histogram comparison doesn't guarantee one-to-one region correspondence. Thus, we associate to each region $A_i \in \mathcal{P}_{S_A}$ a set of regions in $\mathcal{P}_{S_B}$ that provides a good match, so we expect that the correct correspondence is in the initial set of possible correspondences.

Finally, we build the initial set of potential region correspondences as the region pairs characterized by low dissimilarity measures. In the next section we show how this initial potential region correspondences set is constructed.

### Initial set of potential correspondences

The first part of the global registration algorithm aims at finding an initial set of potential region correspondences, $C(A_i)$, for each region $A_i \in S_A$ in the *source* by selecting all regions $\{B_i^l\}$ in the *target* where the dissimilarity is such that:

$$C(A_i) = \{B_i^l \in S_B \mid Ds(A_i, B_i^l) < \varepsilon_{Ds}\} \tag{4.7}$$

where $\varepsilon_{Ds}$ varies in the range $[0, 1]$. The initial correpondence set $C(A_i)$ correspond to all regions in $S_B$ that are distant $\varepsilon_{Ds}$ from $A_i$ in feature space (4.7).

The best value for this threshold depends on the density on both *scans*, which determines the descriptor estimation accuracy. Since we want to decouple the feature estimation problem

from the global registration problem, we fixed the value of $\varepsilon_{Ds}$ as a user-defined parameter. In [HP05] is proposed a statistical method to estimate parameters similar to $\varepsilon_{Ds}$.

To be able to efficiently compare histograms, we will construct a histogram for every region pair $A_i \in S_A$ and $B_i^l \in C(A_i)$ with the same parameters, so the dissimilarity function (4.6) is computed straightfoward.

The number of cells $\beta$ in the histograms $H(A_i)$ and $H(B_i^l)$ is computed using Scott's rule [Sco79], $\beta = 3.49\sigma_f N^{-\frac{1}{3}}$, where $\sigma_f$ is the standard deviation of the $N$ points. We take $N = |A_i| + |B_i^l|$ and $\sigma_f$ is computed over the data represented by $A_i$ and $B_i^l$.

Then, $H(A_i)$, $H(B_i^l)$ are characterized by the $H_{\min}$ and $H_{\max}$ values, defined by:

$$H_{\min} = \min\{f(\mathbf{x})_{\mathbf{x}\in A_i}, f(\mathbf{x})_{\mathbf{x}\in B_i^l}\} \tag{4.8}$$

$$H_{\max} = \max\{f(\mathbf{x})_{\mathbf{x}\in A_i}, f(\mathbf{x})_{\mathbf{x}\in B_i^l}\} \tag{4.9}$$

where $f(\mathbf{x})$ is the shape descriptor at $\mathbf{x}$.

The advantage of using a threshold over exact matching to find the initial correspondence set is twofold. First, when it is considered more than one possible correspondence for each region $A_i$, we ensure that the correct correspondence, if it exists, is in $C(A_i)$. Second, the exact matching approaches assume that all regions in both *source* and *target* have discriminative information and their descriptors are robust. Only when these hypotheses are verified, exact matching approaches guarantee that the correct correspondences are in the initial potential set.

Finally, the correspondence set is represented by the set $C(A_1) \times C(A_2) \times \cdots \times C(A_n)$. The initial set of region correspondences is illustrated in Figure 4.5. Observe that, region matching based only on the descriptor similarity is not enough discriminant to provide uniquely correct correspondences.

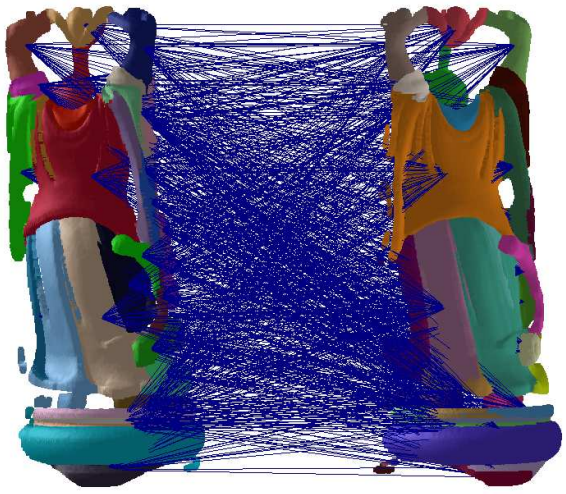


**Figure 4.5:** *Initial set of region correspondences.*

### 4.3.4 Geometric consistency

Given the region set $\mathcal{P}_{\mathbf{S}_A} = \{A_1 \dots A_n\}$ from the *source scan*, let $\{C(A_1), \dots, C(A_n)\}$, with $C(A_i) = \{B_i^l\} \subset \mathcal{P}_{\mathbf{S}_B}$, be the initial set of potential region correspondence, established through region dissimilarity measure (4.3). In this section we will show how we evaluate the geometric consistency to prune the solution space of possible region correspondences.

When evaluating correspondences, the most used point sets distance function is the registration error, also known as *coordinate root mean squared error*, or cRMS (4.1). The cRMS gives a quantitative measure about the correspondence. However, to compute the registration error, the optimal alignment transformation must be computed over the entire dataset. Even for a small number of regions and when each region has a small number potential correspondences, the use of the registration error to find the best set is still prohibitive. We will use the internal distance to evaluate geometric consistency, which is based on the distance between two regions on the same point cloud.

Under rigid transformation assumption, we have that for any region pair $(A_i, B_i^l)$ and $(A_j, B_j^k)$ in correspondence, the pose between the pairs $(\mathbf{c}(A_i), \mathbf{c}(A_j))$ should be the same as between their corresponding regions $(\mathbf{c}(B_i^l), \mathbf{c}(B_j^k))$. The error metric based on inter-point distance is known as *distance root mean squared error*, or dRMS, and it is computed by comparing all internal pairwise distances related to the correspondences:

$$dRMS^2 = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} (\|\mathbf{c}(A_i) - \mathbf{c}(A_j)\| - \|\mathbf{c}(B_i^l) - \mathbf{c}(B_j^k)\|)^2 \tag{4.10}$$

where $n$ is the number of correspondences for the given regions $B_i^l \in C(A_i)$ and $B_j^k \in C(A_j)$. The dRMS function is used as an alternative distance function in [GMGP05] to robustly find a set of corresponding features. They proved the equivalence between the dRMS and the cRMS functions, by showing that dRMS is both lower and upper bounded by cRMS.

The main advantage of using dRMS to evaluate potential correspondences is that it does not require computation of the alignment transformation. In fact, dRMS cost is only computed once for every region pair, since it compares intrinsic properties of two sets of corresponding region, namely the internal pairwise distances, of each region set. For every region pair $(A_i, A_j) \in \mathcal{P}_{\mathbf{S}_A} \times \mathcal{P}_{\mathbf{S}_A}$, the internal pairwise distance is computed only once and stored into a matrix. Independently, the same procedure is repeated for every region pair $(B_i, B_j) \in \mathcal{P}_{\mathbf{S}_B} \times \mathcal{P}_{\mathbf{S}_B}$. When computing dRMS for a set of potential correspondences, the terms in (4.10) corresponding to the internal pairwise distance between two regions are obtained by accessing the respective matrix.

We will use the dRMS distance to evaluate potential corresponding regions. When employed to compare potential feature correspondences, dRMS distance is upper bounded by feature neighborhood size as defined in [GMGP05], and this hypothesis is used to filter wrong correspondences. The same assumption cannot be made when employing dRMS error to compare region correspondences. Even if the segmentation of the two *scans* produces equivalent regions, most of the time, some differences can be observed in the size and shape of the region. As a consequence, it occurs a shift of the region barycenter, and consequently, its position $\mathbf{c}(A_i)$. It

causes a bias on dRMS function cost and this bias cannot be measured or estimated. We assume that we have an one-to-one set correspondences in the overlapping area. If this constraint holds true, we can upper bound the dRMS distance cost to filter wrong correspondence.

### 4.3.5 Global registration algorithm

In this section we will present a branch-and-bound algorithm that searches efficiently the set of potential region correspondence in order to find a triplet of corresponding regions that minimizes the registration error (4.1). The input of the algorithm is the set of regions, where each region is represented by (4.2). Finding the best triplet of corresponding regions is done by descending the decision tree [GLP87]. Each node of the tree describes a potential correspondence between regions in *source* with regions in the *target scan* and the node with the best possible pairing, as measured by a matching function, is chosen as the result of the search. It incrementally makes choices about which regions should be in correspondence, until it converges to the triplet that best registers the *source* with respect to the *target*. The advantage of this search method is that it explore all possible correspondence between *source* and *target* regions, and are guaranteed to find the best possible alignment globally.

The initial set of potential correspondences $\{C(A_0), C(A_1), \cdots, C(A_n)\}$ is computed from (4.7). If any two region pairs $(A_i, B_i^o)$ and $(A_j, B_j^p)$ form a correspondence, they must preserve inter-point distance (4.10), and we have that

$$|\|\mathbf{c}(A_i) - \mathbf{c}(A_j)\| - \|\mathbf{c}(B_i^o) - \mathbf{c}(B_j^p)\|| \approx 0 \tag{4.11}$$

However, region location, represented by $\mathbf{c}(R)$, is shifted when too fine or too coarse segmentation and missing data occurs. In this situation, the correspondences are approximated within a certain error tolerance $\varepsilon_{dRMS}$:

$$|\|\mathbf{c}(A_i) - \mathbf{c}(A_j)\| - \|\mathbf{c}(B_i^o) - \mathbf{c}(B_j^p)\|| < \varepsilon_{dRMS} \tag{4.12}$$

This thresholding (4.12) is applied to verify the geometric consistency between every pair of corresponding regions and it results in a filtering of wrong correspondences. The output of this initial bound is a set of pairs of corresponding regions $\{(A_i, B_i^o), (A_j, B_j^p)\}$, where, for each pair $(A_i, B_i^o), (A_j, B_j^p)$, we have an associated dRMS error (4.10) and a region dissimilarity measure (4.6). The thresholds in branching, $\varepsilon_{dRMS}$ and $\varepsilon_{Ds}$, play an important roles in the performance of our pairwise matching procedure. If they are too large, then the effect of correspondence pruning is not sufficiently strong. Oppositely, if they are too small, we may discard too many correct region correspondences such that there are not enough regions for the matching. A similar thresholding strategy is adopted by feature-based global registration algorithms [SMS$^+$04, GMGP05, HP05], and, all of them consider the theshold value as an user-defined parameter.

The elements of $\{(A_i, B_i^o), (A_j, B_j^p)\}$ are taken as nodes in the decision tree. We construct this tree by connecting the nodes, where each connection gives a possible correspondence set. Given the initial potential correspondence set $\{(A_i, A_j), (B_i^o, B_j^p)\}$ we traverse this set to find

a third correspondence, $(A_k, B_k^q)$ for each pair $\{(A_i, B_i^o), (A_j, B_j^p)\}$. This third correspondence must verify the following:

- The correspondence pairs $\big((A_i, A_k), (B_i^o, B_k^q)\big)$ and $\Big((A_k, A_j), (B_k^q, B_j^p)\Big)$ are potential correspondences.
- And $A_i \neq A_k$, $A_j \neq A_k$, $B_i^o \neq B_k^q$ and $B_j^p \neq B_k^q$.

If these properties are verified, then their respective nodes in the decision tree are connected. This procedure limitates the potential correspondences to the set of nodes connected in the decision tree. After this bounding, we obtain the set of potential triplets of corresponding regions. Associated to every triplet we have the dRMS error (4.10) and the region dissimilarity measure (4.6). This list of potential correspondences is typically smaller than the preceding set. To decrease even more the number of potential correspondence, we sort this set in order of increasing dRMS cost following by a pruning process and repeat the same procedure using the region dissimilarity measure. At this point, the set of potential correspondences is smaller and consistent, compared to the preceding stages. We can then afford to use a more expensive procedure to find the best correspondence triplet.

Once we have a small but consistent set of potential triplet correspondences, we need to find the triplet that minimizes the distance between the scans after the initial alignment. This is done by evaluating the cRMS error after the initial alignment. Given a potential triplet correspondence $\Big((A_i, A_j, A_k), (B_i^o, B_j^p, B_k^q)\Big)$, we take the corresponding points $(\mathbf{c}(A_i), \mathbf{c}(B_i^o))$, $(\mathbf{c}(A_j), \mathbf{c}(B_j^p))$ and $(\mathbf{c}(A_k), \mathbf{c}(B_k^q))$ to estimate the coarse alignment transformation. We refine this estimate by applying two iterations of the original ICP algorithm [Zha94] on the entire dataset. We run ICP algorihtm to the entire dataset and it gives the registration error, cRMS, of the alignment. Only a small number of iterations is necessary, once the global registration estimates only the rough *scans* alignment. Finally, the rigid transformation that gives the smallest cRMS error is taken as the solution of the global registration problem.

Generally, our global registration algorithm proceeds as follow:

1. **Build the initial set of potential correspondences:** For each region $A_i$, we will designate the $j$-th member of the potential correspondence set $C(A_i)$ as $B_i^o$, if $Ds(A_i, B_i^o) < \varepsilon_{Ds}$.

2. **Form region pairs:** For each region pair $(A_i, A_j) \in \mathcal{P}_{S_A}$, it is taken the set of corresponding pairs $(B_i^o, B_j^p) \in \mathcal{P}_{S_B}$, where $B_i^o \in C(A_i)$ and $B_j^p \in C(A_j)$, characterized by a dRMS (4.10) less equal than a given threshold $\varepsilon_{dRMS}$. It leads to a set of potential initial correspondences. We sort this set in order of increasing distance discrepancy.

3. **Add a region to form triplets:** From the set of potential corresponding region pairs, we traverse the search space looking for a third correspondence, $(A_k, B_k^q)$. If we do not find a third potential correspondence, we remove the correspondence pair from the potential pair. Otherwise, we find the region pair $(A_k, B_k^q)$ that minimizes the dRMS function.

4. **Prune region triplets:** At the end of step 3, we obtain a set of potential triplets characterized both by a minimum dissimilarity (4.6) and a minimum $dRMS$ error (4.10). A prune process is performed in order to retain triplets that minimize both distance functions. We sort this set in order of increasing $dRMS$ distance discrepancy, followed by a pruning. We repeat the same procedure using the region dissimilarity cost.

5. **Registration test:** After pruning, we apply the alignment transformation and take the triplet corresponding regions with the minimum $cRMS$ associated.

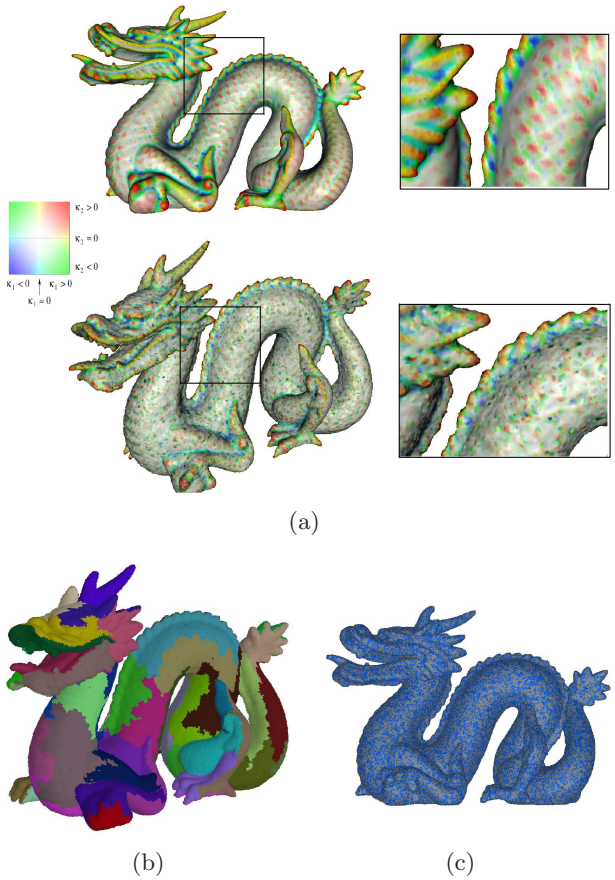
The overall structure of the correspondence search is similar to other global registration algorithms. The basic idea is, at each step of the algorithm, to narrow the set of potential correspondences and to increase the complexity of the operations performed. This is a greedy approach, and, at each step, the set of correspondences decreases, but it strengthens the equivalence condition of resulting potential correspondences. The aligning transformation, the opperation with the biggest cost, is only computed in the last stage. The advantage of this search algorithm is that it explores all possible assignments between regions in both *source* and *data*. In the next section, we will show some experimental results to both illustrate and evaluate the algorithm.

## 4.4 Experimental results

To evaluate the efficiency of our approach, there are two qestions that we need to consider. First, how efficient is our method in practice? And second, how good is the obtained alignment? In this section we present the results of experiments designed to address these two.

We tested our region-based global registration algorithm on a variety of input data with varying amount of noise, outliers, and extent of overlap. We now report performance regarding the robustness to noise and to partial region overlapping. Figure 4.3 shows the main steps of the algorithm and the resulting transform when our algorithm is applied on the Stanford Bunny scans dataset. Although in the example the *source* and *target* point clouds are shown in similar positions, we stress out that algorithm does not depend on any assumptions about the initial positions of the input point clouds.

Figure 4.6 shows the robustness of the region-based global registration algorithm under a zero-mean Gaussian noise without any ICP refinement. We align the Stanford Dragon model to a copy of itself corrupted by zero-mean Gaussian noise. Figure 4.6 (a) shows the *principal curvature* colormap, and it illustrates how noise affects the estimation of geometric descriptors. We set the segmentation partition threshold as $K = 30\%$ for both *source* and *target*. Figure 4.6 (b) shows the resulting regions after segmentation, where the criterion to be optimized in the segmentation was the angle between normals. Despite the noise affects geometric descriptor values, the regions of both *source* and *target* were exactly the same. It shows the repeatability of the segmentation [ADC07a] under noise. In this scenario, each region in both point clouds has the same barycenter, but different histograms. The histogram represented the Gaussian curvate p.d.f. over the region. The initial potential correspondence set was built using $\varepsilon_{Ds} = 0.5$. Our alignment brings the *source* (noisy) point cloud into exact alignment to the *target* (smooth) point cloud (fig. 4.6 (c)).



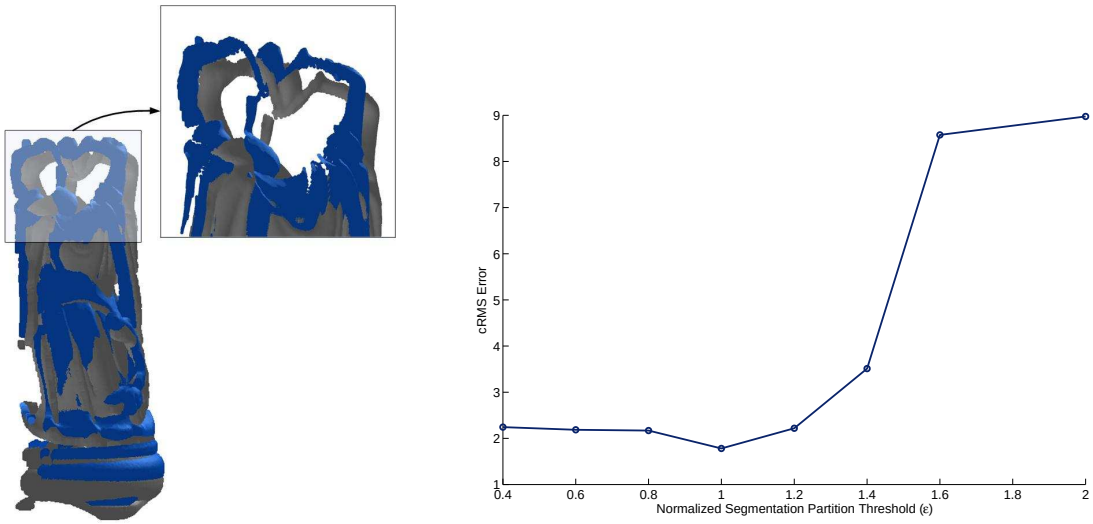
(a)

(b) (c)

**Figure 4.6:** *Dragon example. (a) Input to the region-based global registration algorithm: raw point cloud (the source) and the noisy dragon (the target) with the curvature colormap. (b) Regions obtained after segmentation. (c) Registration after applying our algorithm.*

Figure 4.7 shows the robustness of the region-based global registration algorithm under different segmentations. We align two raw point clouds of the Stanford Happy Buddha model. The magnitude of the segmentation partition $\epsilon$ is varied in a scale where $K = 30\%$ was taken as one unit. Figure 4.5 shows the initial potential correspondence set for $K = 30\%$ .The pose computed by our algorithm is refined by running two iterations of ICP and the alignment error is computed using the cRMS point-point error metric [Zha94]. The results are showed in Figure 4.7(b). For all segmentations, our region-based global registration converges to a correct alignment.

When $\epsilon < 1\%$, the point clouds segmentation is too fine. Thus, the number of regions per point cloud is larger, and, consequently, the solution space of possible region correspondences. The alignment error for over-segmented regions is similar to the error of a correct segmentation. The algorithm robustness to over-segmentation is thanks to the three region correspondence hypothesis. As the regions get small, region histogram descriptor is not enough discriminant and the alignment rely mostly on the geometric constraint. Notice how the alignment error increases when the input point clouds are under-segmented. This is because there are less regions and a smaller solution space of possible correspondences. In this condition, the algorithm only finds the correct alignment because both point clouds have a large overlapping area.




(a) (b)

**Figure 4.7:** *Evaluation of the initial alignment quality under segmentation variation. (a) Initial alignment when $\epsilon = 1$. Detail shows the error in the alignment. (b) Graph of cRMS error as the function of the segmentation partition threshold $\epsilon$.*

## 4.5 Conclusions

In this Chapter, we have presented an algorithm that solves the alignment problem using regions correspondence for two *scans* without any assumption about their initial position. The main contribution of the work is to formulate the global registration under a different perspective, considering region, instead of feature points, as the point cloud representation. We match regions according to their descriptors signatures, so the cost to establish these matches is much cheaper than feature strategies [PMW05], which require feature robust estimation (usually in several scales) and selection.

Experimental results on raw point clouds have illustrated some properties of the proposed global registration scheme. The algorihtm estimated correctly the coarse alignment for *scans* with large overlapping area, which the hypothesis of three equivalent regions was verified. Besides, it is robust to noise.

We believe that the main weakness of our approach is that it makes the strong assumption that at least three one-to-one region equivalence is present in the overlapping area. Although this assumption holds true for many examples, the algorithm failed to converge toward a correct alignment when only partial region matching was present in the overlapping area and, in practice, the power of a single region descriptor is limited. One improvement to solve this problem would be to consider the problem of matching regions under unstable segmentations. It would not only improve the robustness of the algorithm to too fine and too coarse segmentation, but it would also make it.

# Chapter 5

# Conclusions and future directions

> "The visions we offer our children shape the future. It matters what those visions are. Often they become selfulfilling prophecies. Dreams are maps."
>
> - Carl Sagan (Pale Blue Dot)

In this dissertation, we have presented some fundamental building blocks for processing shape information represented as unstructured point clouds, and demonstrated the utility of reducing the data volume through segmentation in a region-based global registration scheme. We conclude with a discussion of some key points developed in this thesis and some extensions and improvements to the algorithms proposed in this thesis to serve as a potential starting point for future work in this area.

We propose the use of raw point cloud as a unified modelling primitive for acquisition, registration, integration and rendering of 3D objects. In the context of processing the *scans* in a modelling from reality pipeline, the main contributions of this thesis are:

- Algorithms for estimating the curvature at each point directly on raw point clouds.
- A graph-based segmentation algorithm designed to partition large, dense, probably with holes, raw point clouds.
- A mathematical formulation for the $MST$-based segmentation algorithm.
- A mathematical formulation of the regions obtained from the segmentation.
- A taxonomy of 3D segmentation algorithms according to the criterion function they aim at minimizing and to the optimization technique used to perform such minimization.
- An algorithm to search for region triplets correspondence, in the context of global registration.

- Evaluation of the proposed algorithms in both synthetic and raw point clouds to validate and outline the properties of each approach.
- Experimental comparison, when possible, of our algorithms with some of the state of arts approaches.

We have showed that with careful application, point cloud segmentation can be an effective mechanism for reducing the data volume. In fact, the lessons learned from the segmentation experiments motivated us to represent the raw point cloud as a set of regions, and use this feature space to perform registration on large datasets. In the following sections, we will present the main conclusions of each problem dealt in this thesis, and some thoughts about extensions of this work and future research.

## 5.1 Local shape descriptors estimation

We began by asking whether local shape descriptors could be used to characterize a raw point cloud. To answer that question, we extended two state-of-the-art curvature estimation methods to work directly on raw point clouds. We proved that both approaches have equivalent *direction curvature* formulation but different discrete approximation to the *principal curvatures*. We use a geometric graph, called *neighborhood graph*, to structure the dataset. Our experiment results showed that the curvature estimation using these approaches is highly sensitive to noise, even for a small noise magnitude. It provided accurate estimation only for uniform sampling, noiseless, and low curvature point clouds.

Recently, new local shape descriptors have been proposed. For example, the integral invariant [PWHY09], is a low dimensional descriptor that embeds curvature information and is estimated though an integral approach, instead of differential. Although it leads to more stable estimation, this method is only robust to low level noise, uniform sampling, and all applications using this descriptor perform data smoothing and resampling before estimating local shape descriptors. Shape descriptors that are robust to non-uniform sampling and noise, are usually high dimensional, and thus, impossible to be used to describe large datasets. We then conclude that none of these local shape descriptors are enough robust to be used to raw point cloud representation and we believe that the problem of estimating robustly local shape descriptors of large raw point clouds is still an unsolved problem and it deserves a deeper study.

We have also examine the normal estimation algorithm proposed by Hoppe in [HDD$^+$92]. We have found that the normal estimation using this plane fitting approach provides an efficient, and robust, raw point cloud normal estimation. The accuracy of the estimation is dependent on the neighborhood size considered. An adaptive neighborhood, that estimates the optimal neighborhood according to local noise, curvature, and density, provides a more accurate estimation than a unique global neighborhood size. However, since the use of a unique neighborhood size still provides acceptable results, for a large range of neighborhood sizes, we highly recommend the use of this last approach to estimate the normals of raw point clouds, due to its efficiency. The problem that remains unsolved is the global orientation of normals, when the scan is represented by disjoint regions. An approach that uses a global function, such as the ones used in

[LB07, ACSTD07], to evaluate the global orientation consistency should be used to orient these normals.

We have showed that the estimation normals at a point are more accurate and more robust than the curvature descriptors. This fact has motivated the use of normals, or normal based descriptors, in the following stages of the modelling from reality pipeline. The precaution that must be taken is that in some stages of the pipeline the local descriptor must be invariant under rigid transformation, and the direct use of the normals is prohibitive. In these cases, we use the normal cone as our local shape descriptor.

## 5.2 Raw point cloud segmentation

Region obtained from segmentation is one possible data-driven mechanism for grouping points. In this thesis, we have examined the issues related to point cloud segmentation. The first conclusion is that data-driven segmentation rarely produces regions that agree with the human perception of the *scene*. Thus, we evaluated the segmentation according to the invariability of the result, when a variation is observer either in the input parameters, either in the sampling density, either in the magnitude of the noise. These criteria guide a qualitative evaluation of the segmentation algorithm, but there is still little formal understanding of how they influence the final segmentation. It limits a quantitive comparison between different algorithm and the use of data-driven segmentation results into a higher level application.

We present a mathematical formulation to the $MST$-based segmentation and to the regions obtained from segmentation. This formulation helps to the evaluation of the algorithm, to find the limitation of the approach, and it guides potential improvements.

We also proposed a taxonomy for 3D segmentation algorithms that classify an algorithm according to the criterion function it aims at minimizing and the optimization technique used. However, there are no established benchmarks for data-driven, 3D segmentation algorithms and the comparison between methods in the same category is still done visually. An extension of the segmentation evaluation algorithm [UPH07] to 3D, data-driven, segmentation algorithms, would allow an objective comparison between methods, outline the main advantages and limitation of each methods, and it would guide on the choice of the best segmentation algorithm for a given application.

Another conclusion which arose from the evaluation of our segmentation algorithm was that segmentation is sensitive to the parameters used. Our segmentation algorithm produced segmentations of vastly different qualities depending on the parameters used. In addition, the same parameter choice did not produce segmentations of equal quality on all of point clouds. An improvement to the algorithm would be an automatic parameters tuning. It would be consist in finding the parameters zone where the segmentation is stable. However, such an approach would considerably increase the complexity of the segmentation algorithm, and, depending on the application, such an approach is not necessary. An obvious extension of this work is to use our graph-based segmentation algorithm to partition medical volumetric datasets.

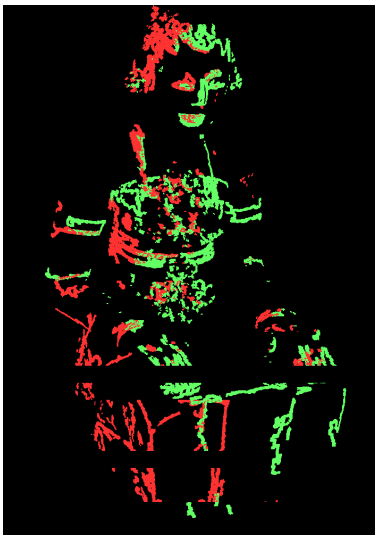
## 5.3 Global registration based on region correspondence

In Chapter 4 we integrate the regions obtained from the point cloud segmentation into a global registration algorithm. Region correspondence is established through histogram similarity between regions in two *scans* and the verification of the geometric consistency between a set of corresponding regions. The algorithm exploits directly the segmentation results. It finds the correct alignment between *scans* when the overlapping area is large, when it is observed more than three regions in correspondence. However, the algorithm performs poorly, converging to a local minima or diverging, when partial matching and small overlapping occurs. A better understanding of shape similarity and formalizing the concept of a distance metric for shapes will help us to better understand the shape space. An improvement would be to make the hypothesis of partial region matching, and use more robust region descriptors and similarity function. Partial region matching has already been considered in the literature for 2D images [VH08], and in 3D point clouds [WWJ+07] and the integration of this concept would improve considerably the robustness of the global registration algorithm. Further research needs to be conducted in this area with rigorous testing on *scans* of various quality and resolution to evaluate the benefits of such methods.

In our global registration algorithm, the choice of the *target* and the *source scan* plays an important role, since the correspondence is not reflexive. Intuitively, the most appropriated *source* would be the one with a larger number of discriminative regions. One way to analyse if a region is discriminative is through the Principal Component Analysis of the region dissimilarity matrix (4.6). We use the Principal Component Analysis (PCA) to estimate the projection that best represents the dissimilarity matrix in a least-squares sense [DHS00]. PCA seeks directions that are efficient for discrimination. Given the dissimilarity matrix, we compute the PCA and we look at the eigenvalues. The eigenvalue amplitude indicates data variation on the direction of the eigenvector. For large eigenvalues, we have a larger disparity of data on the direction of the associated eigenvector, the opposite is verified to small eigenvalues.

Intuitively, if the eigenvalues have similar values, the data distribution is homeomorphic to an hypersphere. It means that regions are not enough discriminant and the initial potential correspondence set does not give enough constraints to guarantee the convergence of the algorithm towards a correct result. The idea is then to choose as the data the point cloud with the most discriminant matrix.

Another possible improvement would be to relax the hypothesis of minimum three region pairs correspondence and formulate the global registration as the problem of finding the complete overlapping area between two *scans*. One solution is to consider this correspondence problem an energy minimization solved through partition and use a graph based optimization technique, such as the a global minimization via graph cut [BK04], or using Region Adjacency Graphs to perform a graph matching.



(a)

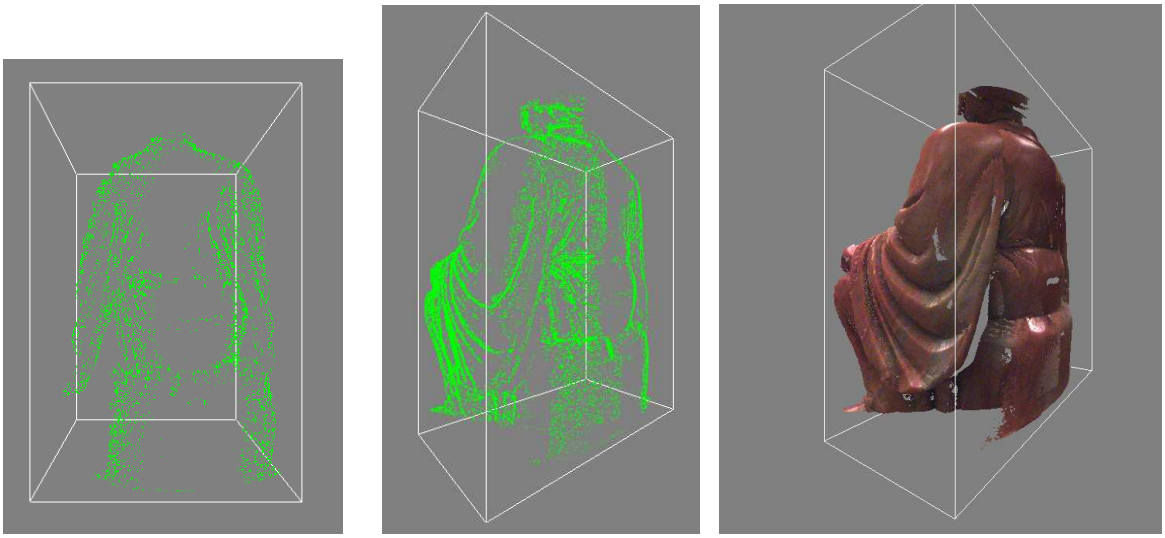
**Figure 5.1:** *Another example of the ICP fine registration alignment when only boundary points are used.*

## 5.4 Region-based point sampling for registration

The regions obtained from segmentation can be integrated to other higher level applications. Here, we will show briefly how regions can be used to improve local registration convergence rate.

The ICP is a non-linear local search algorithm, and it suffers from many problems commonly associated with local searches, such as slow convergence and the tendency to fall into local minima. The point selection strategy and the choice of error metric to be minimized play a large role in both the rate of convergence and the accuracy of the resulting pose. Sampling algorithms [GIRL03, RL01] have showed that discriminant features constraint mainly translation while non discriminant feature constraints mainly rotation.

Based on these considerations we can use segmentation results to guide a point sampling algorithm. The region-based point sampling algorithm reduces the data volume at the same time choose feature points to perform the local registration ICP. Point sampling determines the registration quality and it is crucial to large datasets registration. Point sampling has been addressed from the perspective of convergence rate and stability of local registration in [GIRL03, RL01]. We present an alternative point sampling using regions and boundary properties to select points that best constraint the estimation of the rigid transformation parameters. Figure 5.2 illustrate the principle of this point sampling algorithm. Given a region set, obtained from segmentation, we select all points belonging to the *cuts* and we randomly pick some amount of region points. An example of a reduced point cloud is showed in Figure 5.2(a), and it consists of 15% of the input data. The ICP local registration is applied to the reduced dataset. In these first experiments, we observed that the convergence is faster with the reduced data and the final ICP error is equivalent. This is an example of a potential application where the regions obtained from segmentation can be used in a higher level application.



(a) (b) (c)

**Figure 5.2:** *Fine registration example. Points from the boundary and random region points are taken as a reduced representation for the point clouds (a). The ICP performs the local registration on these reduced datasets (b) and the final alignment is showed in (c).*

# Bibliography


[3DR] 3DReshaper. `http://www.3dreshaper.com/`.

[AB94] R. Adams and L. Bischof. Seeded region growing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(6):641–647, 1994.

[AB98] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. In *SCG '98: Proceedings of the fourteenth annual symposium on Computational geometry*, pages 39–48, New York, NY, USA, 1998. ACM.

[ABCO+03] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and Rendering Point Set Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[ACA07] Rémi Allègre, Raphaëlle Chaine, and Samir Akkouche. A streaming algorithm for surface reconstruction. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 79–88, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[ACSTD07] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 39–48, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[ADC07a] Carla S. R. Aguiar, Sébastien Druon, and André Crosnier. 3D datasets segmentation based on local attribute variation. *International Conference on Intelligent Robots and Systems*, pages 571–578, 2007.

[ADC07b] Carla Silva Rocha Aguiar, Sébastien Druon, and André Crosnier. Hierarchical segmentation for unstructured and unfiltered range images. *Computer Graphics, Imaging and Visualization, International Conference on*, 0:261–267, 2007.

[ADC09] Carla S. R. Aguiar, Sébastien Druon, and André Crosnier. Pairwise region-based scan alignment. *International Conference on Intelligent Robots and Systems*, 2009.

[AK04] Nina Amenta and Yong Joo Kil. Defining point-set surfaces. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 264–270, New York, NY, USA, 2004. ACM.

[AKpKS99] Mihael Ankerst, Gabi Kastenmller, Hans peter Kriegel, and Thomas Seidl. 3D shape histograms for similarity search and classification in spatial databases. pages 207–226. Springer, 1999.

[AM96] A. J. Abrantes and J. S. Marques. A class of constrained clustering algorithms for object boundary extraction. *IEEE Transactions on Image Processing*, 5(11):1507–1521, November 1996.

[AM98] L. Ambrosio and C. Mantegazza. Curvature and distance function from a manifold. *J. Geom. Anal.*, 8:723–748, 1998.

[AMCO08] D. Aiger, N. J. Mitra, and D. Cohen-Or. 4-points congruent sets for robust surface registration. *ACM Transactions on Graphics*, 27(3), 2008.

[APFJ+08a] M. Alcoverro, S. Philipp-Foliguet, M. Jordan, L. Najman, and J. Cousty. Region-based 3d artwork indexing and classification. In *Proceedings of the IEEE 3DTV-Con Conference*, Istanbul, May 28-30 2008.

[APFJ+08b] M. Alcoverro, S. Philipp-Foliguet, M. Jordan, L. Najman, and J. Cousty. Region-based 3d artwork indexing and classification. In *Proceedings of the IEEE 3DTV-Con Conference*, Istanbul, May 28-30 2008.

[AR05] M.s. Bae A. Razdan. Curvature estimation scheme for triangle meshes using bi-quadratic bezier patches. *Computer Aided Design*, 37(14):1481–1491, 2005.

[BAAC08] B. Ben Amor, M. Ardabilian, and Liming Chen. Toward a region-based 3d face recognition approach. In *Multimedia and Expo, 2008 IEEE International Conference on*, pages 101–104, June 23 2008-April 26 2008.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[Bes88] Paul J. Besl. *Surfaces in range image understanding.* Springer-Verlag New York, Inc., New York, NY, USA, 1988.

[BFL06] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient n-d image segmentation. *Int. J. Comput. Vision*, 70(2):109–131, 2006.

[BJ88] P. J. Besl and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(2):167–192, 1988.

[BK04] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, 2004.

[BM92] P.J. Besl and N.D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[BM93] S. Beucher and F. Meyer. *The morphological approach to segmentation: the watershed transformation.* Marcel Dekker, 1993.

[BMG94] Kim L. Boyer, Muhammad J. Mirza, and Gopa Ganguly. The robust sequential estimator: A general approach and its application to surface organization in range data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(10):987–1001, 1994.

[BR07] Benedict Brown and Szymon Rusinkiewicz. Global non-rigid alignment of 3-D scans. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3), aug 2007.

[BRM+02] Fausto Bernardini, Holly Rushmeier, Ioana M. Martin, Joshua Mittleman, and Gabriel Taubin. Building a digital model of michelangelo's florentine piet'a. *IEEE Computer Graphics and Applications*, 22:59–67, 2002.

[Bro92] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, 1992.

[Bro08] Benedict J. Brown. *Registration and Matching of Large Geometric Datasets for Cultural Heritage Applications.* PhD thesis, Princeton University, June 2008.

[BS02] O. R. P. Bellon and L. Silva. New improvements to range image segmentation by edge detection. *IEEE Signal Processing Letters*, 9(2):43–45, February 2002.

[BSM05] Ilja N. Bronshtein, Konstantin A. Semendyayev, and Gerhard Musiol Heiner Muehlig. *Handbook of Mathematics.* Springer-Verlag New York, Inc., New York, NY, USA, 2005.

[BTFN+08] Benedict J. Brown, Corey Toler-Franklin, Diego Nehab, Michael Burns, David Dobkin, Andreas Vlachopoulos, Christos Doumas, Szymon Rusinkiewicz, and Tim Weyrich. A system for high-volume acquisition and matching of fresco fragments: Reassembling Theran wall paintings. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 27(3), August 2008.

[CBV98] R. Chaine, S. Bouakaz, and D. Vandorpe. A graph-anisotropic approach to 3-d data segmentation. In *Computer Graphics, Image Processing, and Vision, 1998. Proceedings. SIBGRAPI '98. International Symposium on*, pages 262–269, Rio de Janeiro, Brazil, October 1998.

[CF01] Richard J. Campbell and Patrick J. Flynn. A survey of free-form object representation and recognition techniques. *Comput. Vis. Image Underst.*, 81(2):166–210, 2001.

[CGR+04] U. Clarenz, M. Griebel, M. Rumpf, M. A. Schweitzer, and A. Telea. Feature sensitive multiscale editing on surfaces. *Vis. Comput.*, 20(5):329–343, 2004.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH '96: Proceedings of the 23rd annual conference*

*on Computer graphics and interactive techniques*, pages 303–312, New York, NY, USA, 1996. ACM.

[CM92] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, 1992.

[CM02] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[COC] COCONE. `www.cse.ohio-state.edu/~tamaldey`.

[Con] ARC Automatic Reconstruction Conduit. `http://www.arc3d.be/`.

[CP03] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 177–187, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[CS92] Xin CHEN and Francis SCHMITT. Intrinsic surface properties from surface triangulation. In *2nd Europeen Conference on Computer Vision-ECCV'92*, pages 739–743, 1992.

[CW00] Péter Csákány and Andrew M. Wallace. Computation of local differential parameters on irregular meshes. In *Proceedings of the 9th IMA Conference on the Mathematics of Surfaces*, pages 19–33, London, UK, 2000. Springer-Verlag.

[CZ05] Miguel Á. Carreira-Perpiñán and Richard S. Zemel. Proximity graphs for clustering and manifold learning. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 225–232. MIT Press, Cambridge, MA, 2005.

[DAa06] Sébastien Druon, M.J. Aldon, and André Crosnier and. Color constrained icp for registration of large unstructured 3d color data sets. *Information Acquisition, IEEE International Conference*, pages 249–255, 2006.

[Dat] Spacetime Face Data. `http://grail.cs.washington.edu/software-data/stfaces/index.html`.

[dC76] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Englewood Cliffs, NJ, 1976.

[Dey06] Tamal. K. Dey. *Curve and Surface Reconstruction : Algorithms with Mathematical Analysis.* Cambridge University Press, 2006.

[DG04] Tamal K. Dey and Samrat Goswami. Provable surface reconstruction from noisy samples. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 330–339, New York, NY, USA, 2004. ACM.

[DGG03] Tamal Dey, Joachim Giesen, and Samrat Goswami. Shape segmentation and matching with flow discretization. In *In Proc. Workshop on Algorithms and Data Structures*, pages 25–36, 2003.

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. 0-471-05669-3. wiley-Interscience Publication, 2000.

[Dij59] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[DJ97] Chitra Dorai and Anil K. Jain. COSMOS - a representation scheme for 3D free-form objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1115–1130, 1997.

[DLS05] T. K. Dey, G. Li, and J. Sun. Normal estimation for point clouds: a comparison study for a voronoi based method. In *Point-Based Graphics, 2005. Eurographics/IEEE VGTC Symposium Proceedings*, pages 39–46, 2005.

[DM98] I.L. Dryden and K.V. Mardia. *Statistical Shape Analysis*. John Wiley, Chichester, 1998.

[DMH08] Nicolas Vandapel Daniel Munoz and Martial Hebert. Directional associative markov network for 3-D point cloud classification. In *Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, June 2008.

[DV06] A. Dupuis and P. Vasseur. Image partitionning by grouping and cue integration. *Image and Vision Computing*, 24:1053–1064, 2006.

[ED] Database EROS-3D. `http://eros3d.ensea.fr/`.

[FB04] A. X. Falcão and F. P. G. Bergo. Interactive volume segmentation with differential image foresting transforms. *IEEE Trans. on Medical Imaging*, 23(9):1100–1108, Sep 2004.

[FH04] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004.

[FHK+04] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bulow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, May 2004.

[FJ89] P.J. Flynn and A.K. Jain. On reliable curvature estimation. In *CVPR '89: Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 110–116, Washington, DC, USA, 1989. IEEE Computer Society.

[FKMS05] Thomas Funkhouser, Michael Kazhdan, Patrick Min, and Philip Shilane. Shape-based retrieval and analysis of 3d models. *Commun. ACM*, 48(6):58–64, 2005.

[fRotMoF] C2RMF (Center for Research and Restoration of the Museums of France). `http://www.c2rmf.fr/`.

[FSdAL04] Alexandre X. Falcao, Jorge Stolfi, and Roberto de Alencar Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):19–29, 2004.

[GBBS04] P. F. U. Gotardo, O. R. P. Bellon, K. L. Boyer, and L. Silva. Range image segmentation into planar and quadric surfaces using an improved robust estimator and genetic algorithm. *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, 34(6):2303–2316, December 2004.

[GCG06] Andreas Griesser, Nico Cornelis, and Luc Van Gool. Towards on-line digital doubles. In *Proc. of the 3rd Intl. Symp. on 3D Data Processing, Visualization and Transmission (3DPVT)*, June 2006.

[GCJ+07] D. Gorisse, M. Cord, M. Jordan, S. Philipp-Foliguet, and F. Precioso. 3d content-based retrieval in artwork databases. In *Proceedings of the IEEE 3DTV-Conference*, Kos Island, Greece, May 7-9 2007.

[GCO06] Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph.*, 25(1):130–150, 2006.

[GDH+00] Luc Van Gool, Filip Defoort, Johannes Hug, Gregor Kalberer, Reinhard Koch, Danny Martens, Marc Pollefeys, Marc Proesmans, Maarten Vergauwen, and Alexey Zalesny. Image-based 3D modeling: Modeling from reality. In *Confluence of Computer Vision and Computer Graphics*. Kluwer, 2000.

[GI04] Jack Goldfeather and Victoria Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Trans. Graph.*, 23(1):45–63, 2004.

[GIRL03] Natasha Gelfand, Leslie Ikemoto, Szymon Rusinkiewicz, and Marc Levoy. Geometrically stable sampling for the ICP algorithm. In *Fourth International Conference on 3D Digital Imaging and Modeling (3DIM)*, October 2003.

[GKH08] Daniel Huber Gunhee Kim and Martial Hebert. Segmentation of salient regions in outdoor scenes using imagery and 3-d data. In *IEEE Workshop on Applications of Computer VIsion (WACV08)*. IEEE Computer Society, January 2008.

[GLB01] G. Godin, D. Laurendeau, and R. Bergevin. A method for the registration of attributed range images. *3D Digital Imaging and Modeling, International Conference on*, 0:179, 2001.

[GLP87] W. E. L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(4):469–482, 1987.

[GMGP05] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Proc. Symp. Geom. Processing*, 2005.

[GSC+07] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. Multi-view stereo for community photo collections. In *Proceedings of the 11th International Conference on Computer Vision (ICCV 2007)*, pages 265–270, Rio de Janeiro, Brazil, 2007. IEEE.

[HBK+09] M. Harders, G. Bianchi, B. Knoerlein, , and G. Szkely. Calibration, registration, and synchronization for high precision augmented reality haptics. *Transactions on Visualization and Computer Graphics*, 15(1):138–149, 2009.

[HCH00] Daniel Huber, Owen Carmichael, and Martial Hebert. 3-D map reconstruction from range data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00)*, volume 1, pages 891 – 897, April 2000.

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and WernerStuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.

[HDD+94] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 295–302, New York, NY, USA, 1994. ACM.

[HFG+06] Qi-Xing Huang, Simon Flöry, Natasha Gelfand, Michael Hofer, and Helmut Pottmann. Reassembling fractured objects by geometric matching. *ACM Trans. Graphics*, 25(3):569–578, 2006.

[HH01] Daniel F. Huber and Martial Hebert. Fully automatic registration of multiple 3D data sets. In *IEEE Computer Society Workshop on Computer Vision Beyond the Visible Spectrum(CVBVS 2001)*, December 2001.

[HJBJ+96] Adam Hoover, Gillian Jean-Baptiste, Xiaoyi Jiang, Patrick J. Flynn, Horst Bunke, Dmitry B. Goldgof, Kevin Bowyer, David W. Eggert, Andrew Fitzgibbon, and Robert B. Fisher. An experimental comparison of range image segmentation algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(7):673–689, 1996.

[HLLS01] Gnter Hetzel, Bastian Leibe, Paul Levi, and Bernt Schiele. 3D object recognition from range images using local feature histograms. *cvpr*, 2:394, 2001.

[HM01] Jianbing Huang and Chia-Hsiang Menq. Automatic data segmentation for geometric feature extraction from unorganized 3-D coordinate points. *IEEE Transactions on Robotics and Automation*, 17(3):268–279, June 2001.

[HP05] Qi-Xing Huang and Helmut Pottmann. Automatic and robust multi-view registration. Technical Report 152, Geometry Preprint Series, Vienna Univ.of Technology, December 2005.

[HSKK01] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212, New York, NY, USA, 2001. ACM.

[HT03] Dominique Hulin and Marc Troyanov. Mean curvature and asymptotic volume of small balls. *Am. Math. Mon.*, 110(10):947–950, 2003.

[HTZ04] Feng Han, Zhuowen Tu, and Song-Chun Zhu. Range image segmentation by an effective jump-diffusion method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1138–1153, 2004.

[JBM+00] X. Jiang, K. Bowyer, Y. Morioka, S. Hiura, K. Sato, S. Inokuchi, M. Bock, C. Guerra, R.E. Loke, and J.M.H. du Buf. Some further results of experimental comparison of range image segmentation algorithms. *icpr*, 04:4877, 2000.

[JH97] Andrew Johnson and Martial Hebert. Surface registration by matching oriented points. In *International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 121–128, May 1997.

[JH99] Andrew Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433 – 449, May 1999.

[Jia00] Xiaoyi Jiang. An adaptive contour closure algorithm and its experimental evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1252–1265, 2000.

[JM07] Anupama Jagannathan and Eric. L. Miller. Three-dimensional surface mesh segmentation using curvedness-based region growing approach. *Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2195–2204, December 2007.

[Joh97] Andrew Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[KB04] Leif Kobbelt and Mario Botsch. A survey of point-based techniques in computer graphics. *Computers and Graphics*, 28:801–814, 2004.

[KFR03] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 156–164, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[KL05] D. Koller and M. Levoy. Computer-aided reconstruction and new matches in the forma urbis romae. *Formae Urbis Romae - Nuove Scoperte, Bullettino Della Commissione Archeologica Comunale di Roma*, 2005.

[KS00] Klaus Koster and Michael Spann. MIR: An approach to robust clustering-application to range image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(5):430–444, 2000.

[KSPA01] A. Koschan, Y. Sun, J. Paik, and M. A. Abidi. Robust crease detection and curvature estimation of piecewise smooth surfaces from triangle mesh approximations using normal voting. In *in Proceedings of VECTOR VOTING 229 the International Conference on Computer Vision and Pattern Recognition*, pages 162–167, 2001.

[KV03] A. Kalaiah and A. Varshney. Statistical point geometry. In L. Kobbelt, P. Schroeder, and H. Hoppe, editors, *Eurographics Symposium on Geometry Processing*, pages 107 – 115, June 23 – 25, 2003.

[LB07] Victor Lempitsky and Yuri Boykov. Global optimization for shape fitting. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[LCS06] Kang Li, Xiaodong Wu Danny Z. Chen, and Milan Sonka. Optimal surface segmentation in volumetric images-a graph-theoretic approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(1):119–134, 2006.

[lib] TriMesh library. `http://graphics.stanford.edu/software/trimesh/`.

[LJLC05] Thomas Lewiner, João D. Gomes Jr., Hélio Lopes, and Marcos Craizer. Curvature and torsion estimators based on parametric curve fitting. *Computers and Graphics*, 29(5):641–655, 2005.

[LMA07] Andrew Johnson Yang Cheng Reg Willson Carlos Villalpando Steve Goldberg Andres Huertas Andrew Stein Larry Matthies, Mark Maimone and Anelia Angelova. Computer vision on mars. *International Journal of Computer Vision*, 2007.

[LPC+00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3D scanning of large statues. In *Proceedings of ACM SIGGRAPH 2000*, pages 131–144, July 2000.

[LW85] Marc Levoy and Turner Whitted. The use of points as a display primitive. Technical report, Technical Report 85-022, Computer Science Department, University of North Carolina at Chapel Hill, 1985.

[MAVdF05] Boris Mederos, Nina Amenta, Luiz Velho, and Luiz Henrique de Figueiredo. Surface reconstruction from noisy point clouds. In *SGP '05: Proceedings of the third*

*Eurographics symposium on Geometry processing*, page 53, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[MB05] Jaesik Min and Kevin W. Bowyer. Improved range image segmentation by analyzing surface fit patterns. *Comput. Vis. Image Underst.*, 97(2):242–258, 2005.

[MCAG07] Mohamed-Hamed Mousa, Raphaelle Chaine, Samir Akkouche, and Eric Galin. Efficient spherical harmonics representation of 3d objects. *Computer Graphics and Applications, Pacific Conference on*, 0:248–255, 2007.

[MGGP06] N. J. Mitra, L. Guibas, J. Giesen, and M. Pauly. Probabilistic fingerprints for shapes. In *Symposium on Geometry Processing*, pages 121–130, 2006.

[Mit06] Niloy J. Mitra. *Algorithms for comparing and analyzing three-dimensional geometry.* PhD thesis, Stanford, CA, USA, 2006. Adviser-Leonidas J. Guibas.

[MN03] Niloy J. Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 322–328, New York, NY, USA, 2003. ACM.

[Mon71] Ugo Montanari. On the optimal detection of curves in noisy pictures. *Commun. ACM*, 14(5):335–345, 1971.

[MP02] M. Mortara and G. Patané. Affine-invariant skeleton of 3d shapes. In *SMI '02: Proceedings of the Shape Modeling International 2002 (SMI'02)*, page 245, Washington, DC, USA, 2002. IEEE Computer Society.

[MPB04] Jaesik Min, M. Powell, and K. W. Bowyer. Automated performance evaluation of range image segmentation algorithms. *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, 34(1):263–271, February 2004.

[MW99] A. P. Mangan and R. T. Whitaker. Partitioning 3D surface meshes using watershed segmentation. *Visualization and Computer Graphics, IEEE Transactions on*, 5(4):308–321, 1999.

[MYHS04] S. Manay, A. J. Yezzi, B. W. Hong, and S. Soatto. Integral invariant signatures. *Proc. of the Eur. Conf. on Comp. Vision*, 2004.

[NG04] Gelfand N. and L. Guibas. Shape segmentation using local slippage analysis. In *Proc. Symp. Geom. Processing*, 2004.

[OFCD01] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Matching 3D models with shape distributions. In *SMI '01: Proceedings of the International Conference on Shape Modeling and Applications*, page 154, Washington, DC, USA, 2001. IEEE Computer Society.

[Ohi] Ohio State University repository. `http://sampl.ece.ohio-state.edu/data/motion/tennis`.

[Pan08] Caroline Pantofaru. *Studies in Using Image Segmentation to Improve Object Recognition*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2008.

[Pau03] Mark Pauly. *Point Primitives for Interactive Modeling and Processing of 3D Geometry*. PhD thesis, ETH Zurich, Switzerland, 2003.

[PGK02] Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 163–170, Washington, DC, USA, 2002. IEEE Computer Society.

[PHYH06] Helmut Pottmann, Qi-Xing Huang, Yong-Liang Yang, and Shi-Min Hu. Geometry and convergence analysis of algorithms for registration of 3D shapes. *Int. J. Comput. Vision*, 67(3):277–296, 2006.

[PKA03] D. L. Page, A. F. Koschan, and M. A. Abidi. Perception-based 3d triangle mesh segmentation using fast marching watersheds. In *in Proceedings of the International Conference on Computer Vision and Pattern Recognition, II*, pages 27–32, 2003.

[PKKG03] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, 22(3):641–650, 2003.

[PMW05] B. M. Planitz, A. J. Maeder, and J. A. Williams. The correspondence framework for 3D surface matching algorithms. *Comput. Vis. Image Underst.*, 97(3):347–383, 2005.

[PMW+08] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. Guibas. Discovering structural regularity in 3D geometry. *ACM Transactions on Graphics*, 27(3):#43, 1–11, 2008.

[pr] 3D puzzles repository. `http://www.geometrie.tuwien.ac.at/ig/3dpuzzles.html`.

[Pri57] R. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[Pro] Stanford Digital Forma Urbis Romae Project. `http://formaurbis.stanford.edu/`.

[Pul99] Kari Pulli. Multiview registration for large data sets. *3D Digital Imaging and Modeling, International Conference on*, 0:0160, 1999.

[PWHY09] Helmut Pottmann, Johannes Wallner, Qi-Xing Huang, and Yong-Liang Yang. Integral invariants for robust geometry processing. *Comput. Aided Geom. Des.*, 26(1):37–60, 2009.

[PWY+07] Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, Yu-Kun Lai, and Shi-Min Hu. Principal curvatures from the integral invariant viewpoint. *Comput. Aided Geom. Design*, 24:428–442, 2007.

[Repa] AIM@SHAPE Shape Repository. http://shapes.aim-at-shape.net/.

[Repb] Stanford 3D Scanning Repository. http://graphics.stanford.edu/data/3Dscanrep/.

[RL00] Szymon Rusinkiewicz and Marc Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 343–352, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[RL01] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Proceedings of the Third Intl. Conf. on 3D Digital Imaging and Modeling*, pages 145–152, 2001.

[RM00] Jos B. T. M. Roerdink and Arnold Meijster. The watershed transform: definitions, algorithms and parallelization strategies. *Fundam. Inf.*, 41(1-2):187–228, 2000.

[Rus04] Szymon Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission*, 2004.

[Sap06] Angel Domingo Sappa. Unsupervised contour closure algorithm for range image edge-based segmentation. *IEEE Transactions on Image Processing*, 15(2):377–384, 2006.

[Sch] Image-based 3D Models Archive. http://www.tsi.enst.fr/3dmodels/index0.html.

[Sco79] David W. Scott. On optimal and data-based histograms. *Biometrika.*, 66(3):605–610, 1979.

[Sed02] Robert Sedgewick. *Algorithms in C++: Part 5 Graph Algorithms*. Addison Wesley, third edition, January 2002.

[SH02] I. Shimshoni and E. Hameiri. Estimating the principal curvatures and the darboux frame from real 3-D range data. In *3DPVT02*, pages 258–267, 2002.

[SM00] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[SMS+04] Y. Shan, B. Matei, H. S. Sawhney, R. Kumar, Daniel Huber, and Martial Hebert. Linear model hashing and batch ransac for rapid and accurate object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, June 2004.

[SOS05] R. Sara, I.S. Okatani, and A. Sugimoto. Globally convergent range image registration by graph kernel algorithm. In *3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on*, pages 377–384, 13-16 June 2005.

[Tau91] Gabriel Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(11):1115–1138, 1991.

[Tau95] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 902, Washington, DC, USA, 1995. IEEE Computer Society.

[TKD05] Sun J. Tamal K. Dey, Goswami S. Extremal surface based projections converge and reconstruct with isotopy. Technical report, Department of Computer Science and Engineering -The Ohio State University, April 2005.

[TM98] Chi-Keung Tang and Gérard Medioni. Inference of integrated surface, curve, and junction descriptions from sparse 3d data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(11):1206–1223, 1998.

[TT05] Wai-Shun Tong and Chi-Keung Tang. Robust estimation of adaptive tensors of curvature by tensor voting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):434–449, 2005.

[TTMM04] Wai-Shun Tong, Chi-Keung Tang, Philippos Mordohai, and Gérard Medioni. First order augmentation to tensor voting for boundary inference and multiscale analysis in 3d. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):594–611, 2004.

[TV04] J. W. H. Tangelder and R. C. Veltkamp. A survey of content based 3D shape retrieval methods. In *Shape Modeling Applications, 2004. Proceedings*, pages 145–156, June 2004.

[TV08] Johan W. Tangelder and Remco C. Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia Tools Appl.*, 39(3):441–471, 2008.

[TW04] R. Keiser S. Heinzle S. Scandella M. Gross T. Weyrich, M. Pauly. Post-processing of scanned 3d surface data. In *Eurographics Symposium on Point-based Graphics*, 2004.

[UH07] Ranjith Unnikrishnan and Martial Hebert. Denoising manifold and non-manifold point clouds. In *18th British Machine Vision Conference (BMVC)*, September 2007.

[UH08] Ranjith Unnikrishnan and Martial Hebert. Multi-scale interest regions from unorganized point clouds. In *Workshop on Search in 3D (S3D), IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2008.

[ULVH06a] Ranjith Unnikrishnan, Jean-François Lalonde, Nicolas Vandapel, and Martial Hebert. Scale selection for the analysis of point-sampled curves. In *Third International Symposium on 3D Processing, Visualization and Transmission (3DPVT 2006)*, June 2006.

[ULVH06b] Ranjith Unnikrishnan, Jean-François Lalonde, Nicolas Vandapel, and Martial Hebert. Scale selection for the analysis of point-sampled curves: Extended report. Technical report, Robotics Institute - Carnegie Mellon University, june 2006.

[Unn08] Ranjith Unnikrishnan. *Statistical Approaches to Multi-Scale Point Cloud Processing*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2008.

[UPH07] Ranjith Unnikrishnan, Caroline Pantofaru, and Martial Hebert. Toward objective evaluation of image segmentation algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):929–944, 2007.

[VG06] Maarten Vergauwen and Luc Van Gool. Web-based 3d reconstruction service. *Mach. Vision Appl.*, 17(6):411–426, 2006.

[VH08] Narendra Ahuja Varsha Hedau, Himanshu Arora. Matching Images Under Unstable Segmentation. *Computer Vision and Pattern Recognition*, 2:394, 2008.

[Wei99] Yair Weiss. Segmentation Using Eigenvectors: A Unifying View. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 975, Washington, DC, USA, 1999. IEEE Computer Society.

[WWJ+07] Sen Wang, Yang Wang, Miao Jin, Xianfeng David Gu, and Dimitris Samaras. Conformal geometry and its applications on 3d shape matching, recognition, and stitching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1209–1220, 2007.

[XOU96] Ying Xu, Victor Olman, and Edward C. Uberbacher. A segmentation algorithm for noisy images. In *IJSIS '96: Proceedings of the 1996 IEEE International Joint Symposia on Intelligence and Systems*, page 220, Washington, DC, USA, 1996. IEEE Computer Society.

[YFM01] Yizhou Yu, A. Ferencz, and J. Malik. Extracting objects from range and radiance images. *IEEE Transactions on Visualization and Computer Graphics*, 7(4):351–364, October/December 2001.

[YKS00] M. Gopiy Y, S. Krishnan, and C. T. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. In *Computer Graphics Forum*, volume 19, pages 467–478(12). Blackwell Publishing, 2000.

[YL89] Naokazu Yokoya and Martin D. Levine. Range image segmentation based on differential geometry: A hybrid approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(6):643–649, 1989.

[YLHP06] Yong-Liang Yang, Yu-Kun Lai, Shi-Min Hu, and Helmut Pottmann. Robust principal curvatures on multiple scales. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 223–226, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[YLL+05] Hitoshi Yamauchi, Seungyong Lee, Yunjin Lee, Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Feature sensitive mesh segmentation with mean shift. In *SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005*, pages 238—245, Washington, DC, USA, 2005. IEEE Computer Society.

[Zah71a] C. T. Zahn. Graph theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20:68, 1971.

[Zah71b] Charles T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transaction on Computers*, 20:68–86, 1971.

[ZG08] Gang Zeng and Luc Van Gool. Multi-label image segmentation via point-wise repetition. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2008.

[Zha94] Z. Zhang. Interative point matching for registration of free-form curves and surfaces. *Int. Journal of Computer Vision*, 13(2):119–152, 1994.

[ZIZ99] Dongmei Zhang, Katsushi Ikeuchi, and Copyright Dongmei Zhang. Harmonic shape images: A 3-D free-form surface representation and. In *Its Application in Surface Matching, Ph.D. dissertation, Carnegie Mellon Univ*, 1999.

[ZPKG02] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3D: an interactive system for point-based surface editing. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 322–329, New York, NY, USA, 2002. ACM.

[ZSCS04] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: high resolution capture for modeling and animation. *ACM Trans. Graph.*, 23(3):548–558, 2004.

[ZY96] Song Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multi-band image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:884–900, 1996.

# Appendix A

# Curvature estimation

In this section we will detail the methods presented in this thesis to estimate *principal curvatures*. Taubin in [Tau95] proposed an algorithm to estimate the *principal curvatures* and *principal directions* of triangle mesh surface representation. The algorithm expresses the second fundamental tensor (2.29) at each point as an integral, constructed in time proportional to the neighborhood size. The integral $M_{\mathbf{x}}$ is a matrix equivalent to the three dimension *directional curvature* tensor (2.29) to non-tangent directions. It has the same eigenvectors of $\Pi$, and their eigenvalues are related by a fixed homogeneous linear transformation.

**Approach:** The orthogonal basis of three-dimensional space for the *directional curvature* (2.29) can be obtained by adding the normal vector $\mathbf{n}$ to the basis $\{\mathbf{t}_1, \mathbf{t}_2\}$. It extends the *directional curvature* definition to non-tangent directions and it is given by

$$\kappa_{\mathbf{x}}(\mathbf{t}) = \begin{pmatrix} n \\ t_1 \\ t_2 \end{pmatrix}^t \begin{pmatrix} 0 & 0 & 0 \\ 0 & \kappa^1_{\mathbf{x}} & 0 \\ 0 & 0 & \kappa^2_{\mathbf{x}} \end{pmatrix} \begin{pmatrix} n \\ t_1 \\ t_2 \end{pmatrix} \tag{A.1}$$

where $\mathbf{t} = n\mathbf{n}_{\mathbf{x}} + t_1\mathbf{t}_1 + t_2\mathbf{t}_2$ is an arbitrary vector in which direction the curvature is estimated. To compute the *principal curvatures* from (A.1) we need to first restrict the $3 \times 3$ matrix to the tangent plane to $\Sigma$ at $\mathbf{x}$. It means that the matrix (A.1) must have zeros at the entries of the first column and first row. The eigenvalues and the eigenvectors of the resulting $2 \times 2$ matrix give the *principal curvatures* and *principal directions* of (2.29).

Let $\mathbf{t}_\theta = \cos(\theta)\mathbf{t}_1 + \sin(\theta)\mathbf{t}_2$ be the unit length vector that gives the direction in which the curvature is computed. This vector ¡svaries in the range $-\pi \leq \theta \leq \pi$. From equation (2.29), the *directional curvature* can be written as

$$\kappa_{\mathbf{x}}(\mathbf{t}_\theta) = \kappa^1_{\mathbf{x}} \cos^2(\theta) + \kappa^2_{\mathbf{x}} \sin^2(\theta) \tag{A.2}$$

Taubin uses the symmetric matrix $\mathbf{M}_{\mathbf{x}}$, which is the *directional curvature* integral over all possible directions $\theta$:

$$\mathbf{M}_{\mathbf{x}} = \frac{1}{2\pi} \int_{-\pi}^{+\pi} \kappa_{\mathbf{x}}(\mathbf{t}_\theta)\mathbf{t}_\theta\mathbf{t}_\theta^t d\theta \tag{A.3}$$

Since the unit length normal vector $\mathbf{n_x}$ to $\Sigma$ at $\mathbf{x}$ is an eigenvector of $\mathbf{M_x}$ associated with the eigenvalue zero, it follows that $\mathbf{M_x}$ can be factorized as follows

$$\mathbf{M_x} = \mathbf{T}_{12}^t \begin{pmatrix} m_{\mathbf{x}}^{11} & m_{\mathbf{x}}^{12} \\ m_{\mathbf{x}}^{21} & m_{\mathbf{x}}^{22} \end{pmatrix} \mathbf{T}_{12} \tag{A.4}$$

where $m_{\mathbf{x}}^{12} = m_{\mathbf{x}}^{21} = 0$, $\mathbf{T}_{12} = [\mathbf{t}_1 \mathbf{t}_2]$ concatenates the orthonormal basis of the tangent space, and

$$\kappa_{\mathbf{x}}^1 = 3m_{\mathbf{x}}^{11} - m_{\mathbf{x}}^{22} \tag{A.5}$$

$$\kappa_{\mathbf{x}}^2 = 3m_{\mathbf{x}}^{22} - m_{\mathbf{x}}^{11} \tag{A.6}$$

Taubin proposes a discrete approximation of the integral matrix $\mathbf{M_x}$ (A.3). This matrix is computed at each point, and the discrete matrix $\mathbf{M_x}$ is factorized into (A.4), in order to compute the *principal directions* and *principal curvatures*, using (A.4) and (A.6). The factorization (A.4) avoids the computation of the eigenvectors and eigenvalues of $\mathbf{M_x}$.

**Discrete approximation:** Given a point $\mathbf{x}_i$ and its neighbors $\mathcal{N}_{r_i}(\mathbf{x}_i)$, the matrix $\mathbf{M_x}$ (A.3) can be approximated by a weighted sum over the neighborhood $\mathcal{N}_{r_i}(\mathbf{x}_i)$ as:

$$\hat{\mathbf{M}}_{\mathbf{x}_i} = \sum_{\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)} w_{ij} \kappa_{ij} \mathbf{t}_{ij} \mathbf{t}_{ij}^t \tag{A.7}$$

where $\kappa_{ij}$ is the *directional curvature* when the tangent vector $\mathbf{t}_{ij}$ is the unit length projection of the vector $\mathbf{x}_j - \mathbf{x}_i$ onto the tangent plane. The tangent plane is defined as the plane perpendicular to the normal vector $\mathbf{n}_{\mathbf{x}_i}$. The matrix $\hat{\mathbf{M}}_{\mathbf{x}_i}$ is averaged by the weight $w_{ij}$, which, originally in [Tau95], is proportional to the surface area of all triangles incident to both $\mathbf{x}_j$ and $\mathbf{x}_i$.

The estimated $\hat{\mathbf{M}}_{\mathbf{x}_i}$ is a $3 \times 3$ matrix equivalent to (A.1). As a consequence, we have that the eigenvector associated with the smallest eigenvalue is the normal vector $\mathbf{n}_{\mathbf{x}_i}$. In order to compute the *principal curvatures* from the matrix $\hat{\mathbf{M}}_{\mathbf{x}_i}$, this matrix must be restricted to the tangent plane. To do so, the Householder transformation [BSM05] followed by a diagonalization brings the matrix $\hat{\mathbf{M}}_{\mathbf{x}_i}$ to the tangent plane. The Householder matrix, $\mathbf{Q}_{\mathbf{x}_i}$ is an orthogonal matrix and has its first column equal to $\mathbf{n}_{\mathbf{x}_i}$, or $-\mathbf{n}_{\mathbf{x}_i}$. The two other columns define an orthogonal basis of the tangent space, but not necessarily the *principal directions*. The factorization of $\hat{\mathbf{M}}_{\mathbf{x}}$

$$\mathbf{Q}_{\mathbf{x}_i}^t \hat{\mathbf{M}}_{\mathbf{x}_i} \mathbf{Q}_{\mathbf{x}_i} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \hat{m}_{\mathbf{x}_i^{11}} & \hat{m}_{\mathbf{x}_i^{12}} \\ 0 & \hat{m}_{\mathbf{x}_i^{21}} & \hat{m}_{\mathbf{x}_i^{22}} \end{pmatrix} \tag{A.8}$$

The non-zero entries of (A.8) are the discrete approximation for the $2 \times 2$ matrix in (A.4). Therefore, the *principal curvatures* are estimated using (A.6).

An important observation is that at points where $\lambda_1 = \lambda_2$, the notion of principal direction is not defined, since all vectors are eigenvectors [GI04]. Such a point is called an umbilical point on the surface.

**Estimating $\mathbf{t}_{ij}$ and $\kappa_{ij}$:** Note that, to compute the matrix $\hat{\mathbf{M}}_{\mathbf{x}_i}$ (2.32), we need to estimate, additionally, the tangent vector $\mathbf{t}_{ij}$ and the *directional curvature* $\kappa_{i,j}$ for each pair $(\mathbf{x}_i, \mathbf{x}_j)$. First, the tangent vector $\mathbf{t}_{ij}$ can be computed simply as the projection of the vector $(\mathbf{x}_j - \mathbf{x}_i)$ on the tangent plane $\langle \mathbf{n}_{\mathbf{x}_i} \rangle^{\perp}$:

$$\hat{\mathbf{t}}_{ij} = (\mathbf{x}_i - \mathbf{x}_j) + \beta \cdot \hat{\mathbf{n}}_{\mathbf{x}_i} \tag{A.9}$$

where the constant $\beta$ is given as $\beta = -\hat{\mathbf{n}}^t_{\mathbf{x}_i}(\mathbf{x}_i - \mathbf{x}_j)$ and the normal vector $\hat{\mathbf{n}}_{\mathbf{x}_i}$ is estimated using Hoppe's technique [HDD+92]. The tangent vector $\mathbf{t}_{ij}$ can then be estimated as:

$$\hat{\mathbf{t}}_{ij} = \frac{(\mathbf{I} - \hat{\mathbf{n}}_{\mathbf{x}_i}\hat{\mathbf{n}}^t_{\mathbf{x}_i})(\mathbf{x}_i - \mathbf{x}_j)}{\|(\mathbf{I} - \hat{\mathbf{n}}_{\mathbf{x}_i}\hat{\mathbf{n}}^t_{\mathbf{x}_i})(\mathbf{x}_i - \mathbf{x}_j)\|} \tag{A.10}$$

The second variable that must be computed is the *directional curvature*, $\kappa_{ij}$. The *directional curvature* is estimated from a parametric arc-length curve approximation. Given a *normal section* defined by the tangent vector $\mathbf{t}_{\mathbf{x}_i}$ and the the normal $\mathbf{n}_{\mathbf{x}}$, the curve $\alpha$ parametrized by arc length $s$ (definition 2.28), such that:

$$\alpha(0) = \mathbf{x}_i \tag{A.11}$$
$$\alpha'(0) = \mathbf{t}_{\mathbf{x}_i} \tag{A.12}$$
$$\alpha''(0) = \kappa_{\mathbf{x}_i}(\mathbf{t}_{\mathbf{x}_i})\mathbf{n}_{\mathbf{x}_i} \tag{A.13}$$

The discrete approximation for the arc-length in Laurent series up to the second order gives:

$$\begin{aligned} \alpha(s) &= \alpha(0) + \alpha'(0)s + \frac{1}{2}\alpha''(0)s^2 + O(s^3) \\ &= \mathbf{x}_i + \mathbf{t}_{\mathbf{x}_i}s + \frac{1}{2}\kappa_{\mathbf{x}_i}(\mathbf{t}_{\mathbf{x}_i})\mathbf{n}_{\mathbf{x}_i}s^2 + O(s^3) \end{aligned} \tag{A.14}$$

and we observe that

$$2\mathbf{n}^t_{\mathbf{x}_i}(\alpha(s) - \mathbf{x}_i) = \kappa_{\mathbf{x}_i}(\mathbf{t}_{\mathbf{x}_i})s^2 + O(s^3) \tag{A.15}$$

and

$$\|\alpha(s) - \mathbf{x}_i\|^2 = s^2 + O(s^3) \tag{A.16}$$

From the previous equations, it follows that the *directional curvature* is equal to the limit:

$$\kappa_{\mathbf{x}_i}(\mathbf{t}_{\mathbf{x}_i}) = \lim_{s \to 0} \frac{2\mathbf{n}^t_{\mathbf{x}_i}(\alpha(s) - \mathbf{x}_i)}{\|(\alpha(s) - \mathbf{x}_i)\|^2} \tag{A.17}$$

If $\mathbf{x}_j$ is close from $\mathbf{x}_i$, however different from it, we can approximate the *directional curvature* as

$$\hat{\kappa}_{ij} = \frac{2\hat{\mathbf{n}}^t_{\mathbf{x}_i}(\mathbf{x}_j - \mathbf{x}_i)}{\|(\mathbf{x}_j - \mathbf{x}_i)\|^2} \tag{A.18}$$

The complete *principal curvatures* estimation algorithm proposed by Taubin in [Tau95] and adapted here to work on point clouds is summarized in table A.1.

**Algorithm:** Estimation of *principal curvatures* using Taubin's approach

**Input**: Unorganized point cloud $X = \{\mathbf{x}_i\} \in \mathbb{R}^3$ with $i = 1, \cdots, n$.

1. Construct the *neighborhood graph* $G_{\mathcal{N}}$, for a given sphere radius $r$ (defined through the sphere or $k$-nearest neighborhood system). This graph embeds points spatial connectivity.
2. Estimate the oriented normal vector for each point $\mathbf{x}_i \in X$ using Hoppe's technique [HDD+92] and Johnson orientation propagation algorithm [Joh97]. Note that it is used a different neighborhood size, radius $r$, to estimate such normals.
3. **for** $\mathbf{x}_i \in X$ **do**
4. Initialize the matrix $\hat{\mathbf{M}}_{\mathbf{x}_i} = \mathbf{0}_{3\times 3}$.
5. **for** $\mathbf{x}_j \in N(\mathbf{x}_i)$ **do**
6. Given $\mathbf{x}_i$, $\mathbf{x}_j$ and $\hat{\mathbf{n}}_i$, estimate the curvature $\hat{\kappa}_{ij}$ (2.39).
7. Estimate the tangent vector $\hat{\mathbf{t}}_{ij}$
8. *Compute $w_{ij}$ using (2.40)*
9. $\hat{\mathbf{M}}_{\mathbf{x}_i} = \hat{\mathbf{M}}_{\mathbf{x}_i} + w_{ij}\hat{\kappa}_{ij}\hat{\mathbf{t}}_{ij}\hat{\mathbf{t}}_{ij}^t$ (2.32).
10. **end for**
11. Let $\mathbf{e}_1 = (1, 0, 0)^t$. $\mathbf{Q}_{\mathbf{x}_i} = \mathbf{I} - \mathbf{w}_{\mathbf{x}_i}\mathbf{w}_{\mathbf{x}_i}^t$, where $\mathbf{w}_{\mathbf{x}_i} = \dfrac{\mathbf{e}_1 \pm \hat{\mathbf{n}}_{\mathbf{x}_i}}{\|\mathbf{e}_1 \pm \hat{\mathbf{n}}_{\mathbf{x}_i}\|}$, with a minus sign if $\|\mathbf{e}_1 - \hat{\mathbf{n}}_{\mathbf{x}_i}\| > \|\mathbf{e}_1 + \hat{\mathbf{n}}_{\mathbf{x}_i}\|$ , and plus sign otherwise.
12. Compute $\mathbf{Q}_{\mathbf{x}_i}\hat{\mathbf{M}}_{\mathbf{x}_i}\mathbf{Q}_{\mathbf{x}_i}^t$ to restrict $\hat{\mathbf{M}}_{\mathbf{x}_i}$ to the tangent plane (A.8).
13. Compute the *principal curvatures* $\kappa_{\mathbf{x}}^1$ and $\kappa_{\mathbf{x}}^2$ .
14. **end for**

**Table A.1:** *Estimation of principal curvatures using Taubin's approach.*

### A.0.1 A numerical approximation approach

Flynn *et al.* [FJ89] estimates the *principal curvatures* at a point $\mathbf{x}_i$ by considering the normal variation between $\mathbf{x}_i$ and its neighbors. The strategy adopted is to estimate an approximate of the *directional curvature* for each pair points $\mathbf{x}_i$ and $\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)$ and take the *principal curvatures* as the maximum and minimum *directional curvatures*. It is assumed that the normal vectors are oriented, consistent, and the origin of the reference frame is located at the point of interest and the tangent plane is the one perpendicular to the normal vector.

***Directional curvature* approximation:** From curvature definition (2.28), the *directional curvature* at a point $\mathbf{x}$ in direction of $\alpha'(0) = \mathbf{t}$ can be approximated by:

$$\kappa_{\mathbf{x}}(\mathbf{t}) = \lim_{s \to 0} \frac{\|\alpha'(s) - \alpha'(0)\|}{\|\alpha(s) - \alpha(0)\|} \tag{A.19}$$

where $\alpha(0) = \mathbf{x}$.

Given two close points $\mathbf{x}_i$ and $\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)$, the *directional curvature* (2.42) can be computed as:

$$\hat{\kappa}_{ij} = \frac{\|\mathbf{t}_j - \mathbf{t}_i\|}{\|\mathbf{x}_j - \mathbf{x}_i\|} \tag{A.20}$$

where $\mathbf{t}_j$ and $\mathbf{t}_i$ are the tangent vectors at $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively, and they lie on the curve $\alpha(s)$. Since both tangent vectors and normals have unit length, and $\mathbf{t}_j$ is perpendicular to $\mathbf{n}_j$, such as $\mathbf{t}_i$ to $\mathbf{n}_i$, we have that:

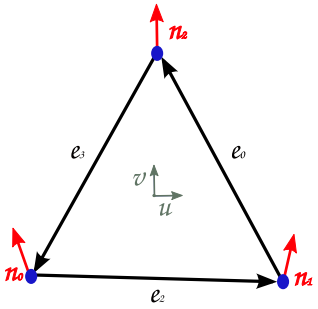
$$\|\mathbf{t}_j - \mathbf{t}_i\| = \|\mathbf{n}_j - \mathbf{n}_i\| \tag{A.21}$$

The *directional curvature* of $\mathbf{x}_i$ in direction of a neighbor $\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)$ is then given as a discrete approximation to one-dimensional curvature along the hypothesized curve from $\mathbf{x}_i$ to $\mathbf{x}_j$ [FJ89]:

$$\hat{\kappa}_{ij} = \begin{cases} \dfrac{\|\mathbf{n}_i - \mathbf{n}_j\|}{\|\mathbf{x}_i - \mathbf{x}_j\|} & \text{if} \quad \|\mathbf{x}_i - \mathbf{x}_j\| \leq \|(\mathbf{n}_i + \mathbf{x}_i) - (\mathbf{n}_j + \mathbf{x}_j)\| \\ -\dfrac{\|\mathbf{n}_i - \mathbf{n}_j\|}{\|\mathbf{x}_i - \mathbf{x}_j\|} & \text{otherwise} \end{cases} \tag{A.22}$$

where the *directional curvature* is positive for locally convex and negative for locally saddle curves.

***Principal curvatures* appoximation:** The algorithm proposed by Flynn does not estimate the actual *principal curvatures*. Instead, the *principal curvatures* are taken as the maximum and the minimum of (2.45) computed for all $\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)$. This is a numerical approximation which is accurate for noiseless, dense and uniformly distributed point clouds. The resulting algorithm is linear, both in time and in space, as a function of the number of vertices and neighborhood size.




$$\mathbf{II}\begin{pmatrix} e_0 \cdot u \\ e_0 \cdot v \end{pmatrix} = \begin{pmatrix} (n_2 - n_1) \cdot u \\ (n_2 - n_1) \cdot v \end{pmatrix}$$
$$\mathbf{II}\begin{pmatrix} e_1 \cdot u \\ e_1 \cdot v \end{pmatrix} = \begin{pmatrix} (n_0 - n_2) \cdot u \\ (n_0 - n_2) \cdot v \end{pmatrix}$$
$$\mathbf{II}\begin{pmatrix} e_2 \cdot u \\ e_2 \cdot v \end{pmatrix} = \begin{pmatrix} (n_1 - n_0) \cdot u \\ (n_1 - n_0) \cdot v \end{pmatrix}$$

**Figure A.1:** *Second fundamental tensor over the tangent plane on a face.* $(u, v)$ *are the directions of an orthonormal coordinate system in the tangent frame and the second tensor is computed from [Rus04].*

Since the method estimates the curvature from differential invariant properties of the surfaces, it amplifies the noise effect. Flynn's curvature estimation algorithm is summarized in table 2.3.

## A.1 Curvature estimation on triangle meshes

We will present for completness Rusinkiewicz's approach.

Rusinkiewicz's approach first estimates the second fundamental tensor over the tangent plane on a face as

$$\Pi = \begin{pmatrix} D_{\mathbf{u}} & D_{\mathbf{v}} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{n}_{\mathbf{x}_i}}{\partial \mathbf{u}} \cdot \mathbf{u} & \frac{\partial \mathbf{n}_{\mathbf{x}_i}}{\partial \mathbf{v}} \cdot \mathbf{u} \\ \frac{\partial \mathbf{n}_{\mathbf{x}_i}}{\partial \mathbf{u}} \cdot \mathbf{v} & \frac{\partial \mathbf{n}_{\mathbf{x}_i}}{\partial \mathbf{v}} \cdot \mathbf{v} \end{pmatrix} \tag{A.23}$$

where $(\mathbf{u}, \mathbf{v})$ is an orthogonal coordinate system in the tangent frame (fig. A.1). Multiplying this tensor by any vector in the tangent plane gives the derivative of the normal in that direction [Rus04]:

$$\Pi \mathbf{s} = D_{\mathbf{s}} \mathbf{n}_{\mathbf{x}_i} \tag{A.24}$$

The second fundamental matrix is estimated from (A.24) at each face, in least square sense, using the relations showed in Figure A.1.

A coordinate system transformation is then performed to express this curvature tensor from the face tangent plane into the vertex coordinate frame. It is computed then an scatter matrix, where the curvature tensor of a vertex is an average of the curvature tensor from adjacent triangles. The eigenvectors and the eigenvalues of this matrix is taken as the *principal directions* and *principal curvatures*, respectively. The algorithm for per-vertex computation of the curvature tensor presented in [Rus04] is showed in table A.3. We estimate the curvature using the software *trimesh* [lib].

**Algorithm:** Estimation of *principal curvatures* using Flynn's curvature approximation

**Input**: Unorganized point cloud $X = \{\mathbf{x}_i\} \in \mathbb{R}^3$ with $i = 1, \cdots, n$.

1. Construct the *neighborhood graph* $G_{\mathcal{N}}$, for a given sphere radius $r$ (defined through the sphere or $k$-nearest neighborhood system). This graph embeds spatial connectivity of points.
2. Estimate the oriented normal vector for each point $\mathbf{x}_i \in X$ using Hoppe's technique [HDD+92] and Johnson orientation propagation algorithm [Joh97]. Note that it is used a different neighborhood size, radius $r$, to estimate such normals.
3. Initialize the *principal curvatures*, $\kappa^1_{\mathbf{x}_i} = -\infty$ and $\kappa^2_{\mathbf{x}_i} = \infty$.
4. **for** $\mathbf{x}_i \in X$ **do**
5. **for** $\mathbf{x}_j \in \mathcal{N}_{r_i}(\mathbf{x}_i)$ **do**
6. **If** $\angle(\mathbf{n}_{\mathbf{x}_i}, \mathbf{n}_{\mathbf{x}_j}) < h$ (2.41)
7. Given $(\mathbf{x}_i, \mathbf{x}_j)$ and $(\hat{\mathbf{n}}_i, \hat{\mathbf{n}}_j)$, compute $\kappa_{ij}$ (2.45).
8. **If** $(\kappa_{ij} > \kappa^1_{\mathbf{x}_i})$
9. **then**
10. $\kappa^1_{\mathbf{x}_i} = \kappa_{ij}$
11. **If** $(\kappa_{ij} < \kappa^2_{\mathbf{x}_i})$
12. **then**
13. $\kappa^2_{\mathbf{x}_i} = \kappa_{ij}$
14. **end if**
15. **end for**
16. Assign to the point $\mathbf{x}_i$ the *principal curvatures*, $\kappa^1_{\mathbf{x}_i}$ and $\kappa^2_{\mathbf{x}_i}$.
17. **end for**

**Table A.2:** *Estimation of principal curvatures using Flynn's curvature approximation.*

**Algorithm:** Estimation of *principal curvatures* using Rusinkiewicz's approach

**Input**: Triangle mesh, with a list of vertices $\mathbf{x}_i \in X$ and edges, $E = \{e\} = \{(\mathbf{x}_i, \mathbf{x}_j)\}$, forming the faces, with a face given by 3 edges $f = (e_1, e_2, e_3)$. Vertex coordinate and normal is also provided as input data.

1. **for** each face **do**
2. Compute edge vectors $e$ and normal differences $\angle(\mathbf{n}'s)$ (Figure A.1).
3. Solve for $\Pi$ using least squares.
4. **for** each vertex $\mathbf{x}_i$ touching the face **do**
5. Re-express $\Pi$ in terms $\mathbf{x}_i$ tangent plane.
6. Add this tensor, weighted by $w_{f,\mathbf{x}_i}$, to vertex curvature. $w_{f,\mathbf{x}_i}$ is takes as the area of face $f$ divided by the squares of the lengths of the two edges that touch vertex $\mathbf{x}_i$.
7. **end for**
8. **end for**
9. **for** each vertex **do**
10. Divide the accumulated $\Pi$ by the sum of the weights
11. The *principal curvatures* and *directions* are taken as the eigenvalues and eigenvectors of $\Pi$.
12. **end for**

**Table A.3:** *Curvature estimation using Rusinkiewicz's approach.*

**Résumé:** La géométrie 3D gagne en popularité en tant que nouvelle forme de contenu multimédia numérique. Il y a un nombre croissant de projets qui ont pour objectif d'acquérir des représentations 3D des objets du patrimoine culturel dans les musées et les fouilles archéologiques. Lacquisition de modèles numériques 3D d'objets du patrimoine culturel permet d'améliorer la conservation, l'archivage, la compréhension, la restauration et la diffusion de ces uvres d'art. Pour atteindre ces objectifs, de nouveaux scanners ont été développés pour capter les détails les plus fins des objets. Les informations fournies par ces systèmes d'acquisition sont constituées de nuages de points bruités, non structurés et fournissant une approximation de la surface continue de l'objet. Les masses de données  traiter sont alors considérables. Dans cette thèse, nous proposons de nouvelles méthodes pour représenter et traiter d'une faon efficace les nuages de points bruités et non structurés. Les travaux réalisés concernent trois axes complémentaires : le traitement des nuages de points pour lextraction de caractéristiques géomtriques (normale, courbure), la segmentation de nuages de points et enfin le recalage basé régions de nuages de points. De nombreuses expérimentations sur des données réelles dobjets dart sont aussi présentées.

**Mots-clés:** géométrie 3D, nuages de points, estimation de normale et courbure, segmentation de nuages de points, recalage basé régions de nuages de points.

**Abstract:** 3D geometry has gained increasing popularity as the new form of digital media content, it has seen an increasing number of projects to acquire detailed 3D representations of cultural heritage objects at museums and archaeological excavations. The goals of acquiring 3D digital models of cultural heritage objects are to improve preservation, archiving, understanding, restoration, and dissemination these artworks. To accomplish these goals, new scanning devices have been developed to capture fine details, irrespective of the size and number of objects, and supplementary information, such as color and texture. The output of these acquisition systems in most of the time is a raw point cloud, a noisy approximation to the continuous object surface, consisting of a set of points coordinates.

The goal of this thesis is to understand and evaluate how large raw point clouds can be efficiently represented and processed in the context of registration. As the volume of data increases, the current registration techniques are not well adapted to treat this problem, and the overload of pre-processing becomes enormous. A compact representation of this data becomes an unavoidable and challenging task. This strategy is appealing because, by focusing attention away from the entire input data to small regions, we place ourselves in a better position to handle practical challenges arising from raw point clouds such as missing data, or occlusion, noise, outliers, and other objects in the scene. We are particularly interested in the development of a registration technique that works directly with the unstructured raw data and with its compact representation, without any other pre-processing.

**Keywords:** discrete geometry, raw point cloud, normal and curvature estimation, raw point cloud segmentation, graph-based segmentation, registration based on region correspondence.