From 3D point clouds to feature preserving meshes

Nader Salman

UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS

ÉCOLE DOCTORALE STIC
Sciences et Technologies de l'Information et de la Communication

THÈSE

pour obtenir le titre de

Docteur en Sciences

de l'Université de Nice-Sophia Antipolis

Mention: Informatique

Présentée et soutenue par

Nader SALMAN

From 3D point clouds to feature preserving meshes

Thèse préparée dans le projet GEOMETRICA, INRIA Sophia Antipolis

Dirigée par
Mariette YVINEC

Soutenue publiquement le 16 Décembre 2010 devant le jury composé de :

Rapporteurs :	Renaud KERIVEN	Professeur	École des Ponts
	Bruno LÉVY	Directeur de Recherche	INRIA
Examinateurs :	Jean-Daniel BOISSONNAT	Directeur de Recherche	INRIA
	Mario BOTSCH	Professor	Bielefeld University
	Paolo CIGNONI	Senior Researcher	ISTI - CNR
	Jean-Philippe PONS	Ingénieur de Recherche	CSTB
Directeur :	Mariette YVINEC	Chargée de Recherche	INRIA

This thesis is

dedicated to my

treasured parents & sister

Résumé

La majorité des algorithmes de reconstruction de surface sont optimisés pour sappliquer à des données de haute qualité. Les résultats obtenus peuvent alors être inutilisables si les données proviennent de solutions d'acquisition bon marché.

Notre première contribution est un algorithme de reconstruction de surfaces à partir de données de stéréo vision. Il combine les informations liées aux points 3D avec les images calibrées afin de combler l'imprécision des données.

L'algorithme construit une collection de triangles 3D á l'aide des images calibrées et l'issue d'une phase de prétraitement du nuage de points. Pour épouser au mieux la surface de la scène, on contraint cette soupe de triangle 3D à respecter des critères de visibilité et de photo-consistance. On calcule ensuite un maillage à partir de la soupe de triangles à l'aide d'une technique de reconstruction qui combine les triangulations de Delaunay contraintes et le raffinement de Delaunay.

Notre seconde contribution est un algorithme qui construit, à partir d'un nuage de points 3D échantillonnés sur une surface, un maillage de surface qui représente fidèlement les arêtes vives. Cet algorithme génère un bon compromis entre précision et complexité du maillage. Dans un premier temps, on extrait une approximation des arêtes vives de la surface sous-jacente à partir du nuage de points. Dans un deuxième temps, on utilise une variante du raffinement de Delaunay pour générer un maillage qui combine les arêtes vives extraites avec une surface implicite obtenue à partir du nuage de points. Notre méthode se révèle flexible, robuste au bruit; cette méthode peut prendre en compte la résolution du maillage ciblé et un champ de taille défini par l'utilisateur.

Nos deux contributions génèrent des résultats efficaces sur une variété de scènes et de modèles. Notre méthode améliore l'etat de l'art en termes de précision.

Abstract

Tremendous progress has been made in recent years in 3D acquisition techniques. These developments triggered a growing demand for increasingly more accurate reconstruction of 3D digital models. However, most of the current algorithms target surface reconstruction from high quality data and can produce some intractable results when used with point clouds acquired through profitable 3D point cloud acquisition methods. Also, over the past three decades both surface reconstruction and mesh generation problems were addressed separately.

The first contribution of this thesis is the development of a simple yet original surface reconstruction algorithm from stereo vision data. Our algorithm copes with the fuzziness of the data by using information from both the multi-view passive stereo extracted 3D point clouds and the calibrated images. After filtering and smoothing the point cloud, the algorithm builds for each calibrated image a triangular depth map respecting the main contours of the image. The triangles of all depth maps are then lifted in 3D space and merged together into a triangular soup. Using a combination of both visibility and photo-consistency constraints, we filter the soup to further enforce it's consistency with the surface of the scene. Finally, a mesh is computed from the triangle soup using a reconstruction method that combines restricted Delaunay triangulation and Delaunay refinement.

Our second contribution is an algorithm that builds, given a 3D point cloud sampled on a surface, an approximating surface mesh. We aim at an accurate representation of surface sharp edges, providing an enhanced trade-off between accuracy and mesh complexity. We first extract from the point cloud an approximation of the sharp edges of the underlying surface. Then a feature preserving variant of a Delaunay refinement process generates a mesh combining a faithful representation of the extracted sharp edges with an implicit surface obtained from the point cloud. The method is shown to be flexible, robust to noise and tunable to adapt to the scale of the targeted mesh and to a user defined sizing field.

Also we demonstrate the effectiveness of both contributions on a variety of scenes and models acquired with different hardware and show results that compare favorably, in terms of accuracy, with the current state of the art.

Acknowledgements

First and foremost, I am deeply indebted to my advisor Mariette Yvinec who has been unstinting in her encouragement and constructive criticism, well beyond the call of duty. Her amazing energy, creativity, uprightness definitively make her an example for me, as a scientist and as a person.

Secondly, I would like to express my gratitude to the French National Research Agency for providing the generous funding that made this research possible. I would also like to acknowledge the distinguished members of my jury: Renaud Keriven, Bruno Lévy, Jean-Daniel Boissonnat, Mario Botsch, Paolo Cignoni and Jean-Philippe Pons. I admire your work and achievements. I am very honored you accepted to review my thesis.

I would also like to express my deep admiration to Pierre Alliez, for being a great and bright scientist always open to discuss new ideas, for having been able to focus me in part of my work, and find a logic in it.

A special thank goes to Laurent Saboret, for his deep programming skills, his sense of humor, his modesty and his great humanity. His infinite kindness gave me hope every time I thought hope was lost. To Anthony Coadou for our inflamed discussions about computer science, philosophy, love and life, for the unconditional and continuous support you still are and for the great honor for choosing me as your best man. I would like to thank Bertrand Pellenard for the *Legen* wait for it... wait for it... *dary* moments we spent during the last couple of months, for his great sense of humor *What uuup?*, his infinite kindness and immense support in the final straight: in one word you were *Awesome!*. To both Anthony Coadou and Bertrand Pellenard for *suiting up* at my Defense. I would like to thank Quentin Mérigot for our scientific collaboration and the numerous discussions and afterwork drinks we shared. I would like to thank Stephane Tayeb for our scientific collaboration and for pushing me to become a home cinema fan. I would like to acknowledge Caroline French for always being kind, supporting and very helpful. I am very grateful to Steve Oudot, Frederic Chazal and Jean-Daniel Boissonnat for giving me the opportunity to visit Stanford University and I am thankful to Leonidas Guibas for welcoming me into the grotto of wonders that was his lab. Thank you to all the Geometrica members for your presence and your support.

This work would not have been possible without the everlasting love and unconditional support of my family. The everlasting attention, love and support of my mother, our invaluable discussions about life, the invaluable support and constant presence of my father, his inspired advice on work, life and uprightness, and our inflamed debates about the world, all this developed my critical mind, my curiosity and my confidence in life. The infinite kindness of my younger sister have always been of major support to me for four of my years here. The wisdom, smartness, love, prayers and care of my grand mother has always been a reassuring presence along my whole life. Together with those who cannot read these words, you made me the person I am today, and you are the main inspiration of my accomplishments.
For all this, I tell you: I love you.

Nader Salman
INRIA Sophia Antipolis
December 2010

Contents

List of Figures

List of Tables

Abbreviations

AABB	Axis Aligned Bounding Box
CCD	Charge-Coupled Device
CVCM	Convolved Voronoi Covariance Matrix
CGAL	Computational Geometry Algorithms Library
DoF	Degree of Freedom
DoG	Difference of Gaussian
FFT	Fast Fourrier Transform
FPE	Feature Preserving Delaunay refinement Extension
k**-D**	k-Dimensional
k**NN**	k Nearest Neighbors
LASER	Light Amplification by Stimulated Emission of Radiation
lfs	local feature size
LoG	Laplacian of Gaussian
LIDAR	Light Detection And Ranging
MLS	Moving Least Squares
NCC	Normalized Cross Correlation
PCA	Principal Component Analysis
PIF	Poisson Implicit Function
PSLG	Planar Straight Line Graph
RBF	Radial Basis Function
SIFT	Scale Invariant Feature Transform
ToF	Time-of-Flight

The gods did not reveal
all things to men at the start;
but as time goes on, by searching,
they discover more and more.
Xenophanes (570-475 B.C.)

1 Introduction

1.1 Motivations

The generation of computer models that can satisfy high modeling and visualization demands is required in different applications, way beyond the well established video-games market. Applications can range from preservation of historical heritage to e-commerce through computer animation, movie production and virtual reality walkthroughs.

Although all the aforementioned applications require some specific processes, they can be classified by their final goal which is visualization, simulation or numerical calculation. One common trend between all these applications is the ever increasing demand for accurate and usable computer model of a physical object which best fits the underlying reality, be it to render high definition images of the object from arbitrary view points under different lighting conditions, or for accurate computations and simulations.

There are two common approaches to create computer models: either the geometry is designed from scratch using some interactive modeling software, or the geometry is digitized from a physical object using some data acquisition hardware and algorithms for reconstructing models from the acquired 3D data. With the current advances in 3D data acquisition hardware, as well as the advances of performance of commodity computers with graphics display capabilities, the acquisition and model reconstruction approach is becoming more and more popular. Indeed, in this thesis we will follow this trend and investigate the topic of surface reconstruction from 3D data acquired from a physical object.

The standard acquisition techniques can be roughly divided into two main categories: acquisition by contact or acquisition without contact. In the first case, the shape is acquired by touching the surface of the object on each relevant side with an adapted instrument. In this thesis we rather focus on the second category of techniques, that is, when data is acquired from the object by means of contact-free methods. More precisely, we are interested in the case were 3D point clouds are derived from laser scanners or by photogrammetric image measurements, *i.e.* multi-view passive stereo.

Early work on surface reconstruction has been focused on data acquired from laser scanners. Although most of these reconstruction methods are reported as successful; the nature of the laser scanners greatly limits their usefulness for the type of scene we are particularly interested in, that is, large scale outdoor scenes. The recent advances in automated multi-view passive stereo matching algorithms can be seen as a valuable alternative for data acquisition. These techniques can produce dense point clouds with a precision getting closer to laser scanner measurements [Seitz et al., 2006]. However, the produced point clouds contain high levels of

noise and lots of outliers. This is principally due to the unavoidable mismatches between images. Therefore, it is often quite difficult to turn the generated point clouds into polygonal meshes of high quality using the standard surface reconstruction methods. As such, these surface reconstruction methods may produce rather incorrect or intractable results from these point clouds. Consequently, it is safe to say that the correct reconstruction method depends on the application first and for each application the right algorithm must be used.

While reconstruction of smooth surfaces from point sets has received a considerable amount of efforts in the past twenty years, adding the piecewise property to the topic is enough to reduce the number of publications by at least an order of magnitude. We can further narrow the taxonomy by distinguishing the work which assumes that the features have been detected from the input 3D point cloud and aim at preserving them, from the ones which recover the features during reconstruction without having to extract them *a priori*. One more step in the narrowing of the previous work can be made by listing the techniques able to achieve piecewise smooth reconstruction from sparse and noisy data sets.

Of course, several techniques for post-processing (smoothing or fairing) the reconstructed surface while preserving the features, be they tagged or inferred, exist. In this thesis we aim at conceiving a reconstruction algorithm which would achieve both feature detection and mesh generation altogether. This issue becomes even more challenging when the input data set is entangled with noise. Furthermore, the feature detection and reconstruction process should take into into account both the scale of the final reconstructed scene and the targeted budget for the size of the output mesh.

This thesis is strongly related to an ANR (Agence Nationale de Recherche or French National Research Agency) project, called Gyroviz[1] (N° ANR-07-AM-013), whose goal is to produce automatic reconstruction of large scale outdoor scenes, typically scenes from cities, from sequences of digital images. The Gyroviz project is an ambitious project carried on by a consortium composed of complementary skills as it is depicted in Figure 1.1. In this project, the camera will be equipped with motion and rotational tracking systems and the resulting information will be used to speed up the stereo vision processing of the images to generate dense 3D point clouds. The subject of this thesis addresses the next step in the reconstruction pipeline, where the set of 3D points, outcome from the multi-view passive stereo processing of images, must be turn into a mesh that is a pertinent and accurate representation of the scene.

1.2 Contributions

As already mentioned, substantial progress has been achieved in data acquisition through multi-view passive stereo techniques. However, only a few reconstruction algorithms tackle successfully the problem of generating a 3D mesh that is a pertinent and accurate representation of the scene from these acquired point clouds. In this thesis we approach this problem in the form of two contributions described hereafter.

The first contribution of this thesis is the development of a simple yet original approach that uses both the multi-view passive stereo extracted 3D point cloud and the calibrated images. Our method is a three-stage process in which, the first stage pre-processes the extracted 3D points through merging, filtering and smoothing. The second stage builds for each calibrated image a triangular depth map respecting the main contours of the image. The triangles of all depth maps are then lifted in 3D and merged together into a triangular soup. We advocate for the combination of both visibility and photo-consistency constraints to further enforce the

[1] http://gyroviz.fr

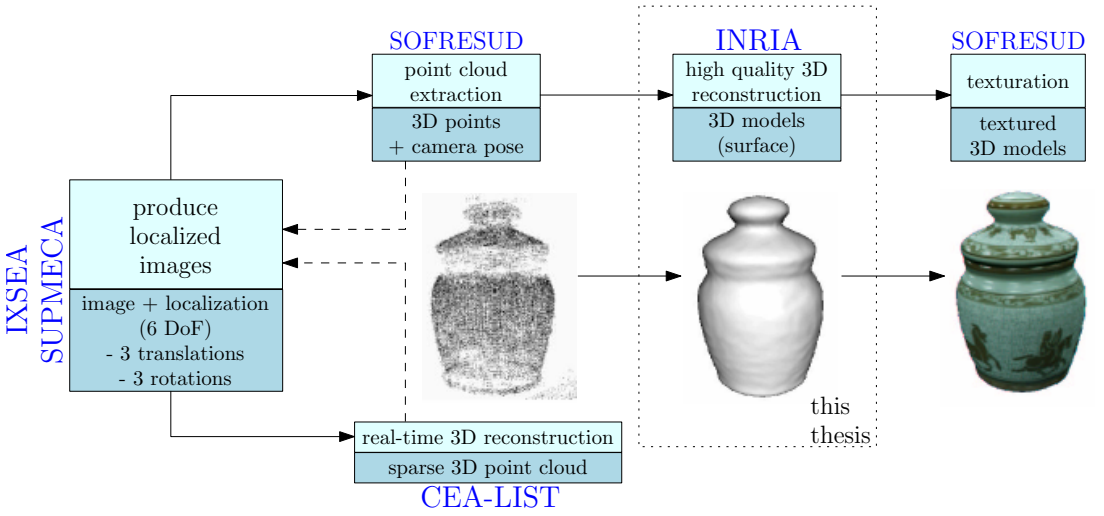


Figure 1.1: The Gyroviz processing line and consortium members (blue).

consistency of the triangle soup with the surface of the scene. Finally, a mesh is computed from the triangle soup using a reconstruction method that combines restricted Delaunay triangulation and Delaunay refinement.

The proposed approach is evaluated through practical experiments and is compared to several other methods: the method is shown to compare favorably with the state of the art when dealing with significantly parched datasets as input and without any post-processing to improve the precision of the resulting meshes. Parts of this work were integrated into a CGAL package described in Appendix A.

Given a potentially noisy 3D point cloud, acquired by laser scanners, sampled on a piecewise smooth surface, our second contribution is an algorithm that builds a mesh approximating the surface with an accurate representation of the surface sharp edges, providing an enhanced trade-off between accuracy and mesh complexity. For that, we base our method on a new robust feature detection process based on the covariance matrices of the Voronoi cells. Our algorithm first extracts from the point cloud an approximation of the sharp edges of the underlying surface. A feature preserving variant of a Delaunay refinement process is then used to generate a mesh combining a faithful representation of the extracted sharp edges with an approximation of the implicit surface obtained from the point cloud by an implicit reconstruction process.

The proposed approach is shown to be flexible, robust to noise and tunable to adapt to the scale of the targeted mesh and to a user defined sizing field. We demonstrate the effectiveness of our new method on a variety of models including real scanned 3D data from indoor and outdoor scenes.

1.3 Outline

The remainder of this thesis is organized as follows:

Chapter 2

In this chapter we give some background on the basic computational geometry concepts men-

tioned throughout this thesis: Delaunay triangulations 2.1, and their weighted counterparts 2.2, Medial axis and ε-sampling 2.3, Haussdorf Distance 2.4, and Restricted Delaunay triangulation 2.5. The reader acquainted with these notions can safely skip this chapter.

Chapter 3

In this chapter we present in more details the input data and we provide practical details on the acquisition techniques that are involved. We start by a quick overview of two common laser range scanning techniques, namely time-of-flight and triangulation systems. Then two different approaches we borrow to compute point clouds from supposedly calibrated images are presented. We also present the goals that the Gyroviz project wishes to attain, as well as the scientific and technical challenges to meet.

Part I: From 3D point clouds to meshes: State of the art

Chapter 4

In this chapter we propose, through a thorough survey, a classification of the different existing surface reconstruction methods from 3D point clouds, with a particular attention on techniques that are viable for datasets entangled with noise and possibly outliers. We also try to highlight their pros and cons regarding the flexibility we wish to attain. The conclusion is drawn that most of the mentioned methods target reconstruction from high quality range data. Consequently there is a need for a surface reconstruction algorithm that is viable for point sets coming from multi-view passive stereo acquisition methods. We recall that these point clouds are entangled with noise and contain lots of outliers.

Chapter 5

In this chapter we propose an elaborate review of existing simplicial mesh generation methods. We start by defining what is a simplicial mesh before describing the three main research directions followed in this domain. The conclusion is drawn that most researchers have seen surface reconstruction and mesh generation as two separate problems. Only few papers combine both. Consequently, combining both surface reconstruction and high quality mesh generation is a research direction we got interested in and that we will expose in Chapter 7.

Part II: From 3D point clouds to meshes: Our contributions

Chapter 6

In this chapter we propose a novel method for reconstructing a 3D model from a point cloud acquired from a set of calibrated images of a large scale outdoor scene. We start by giving some background on the Delaunay refinement surface mesh generation method we employ, we describe our method for multi-view reconstruction from calibrated images and we report some numerical experiments which demonstrate that our results compare favorably with the current state of the art.

Chapter 7

In this chapter we present a novel method for computing a feature preserving mesh from 3D point clouds. We start by presenting the evolution of the literature from smooth to piecewise smooth surface reconstruction before presenting the few mesh generation algorithms that combine surface reconstruction and mesh generation while preserving sharp features. Then, we describe our method for point clouds possibly containing noise. Finally, we report some

numerical experiments which demonstrate the effectiveness, robustness and flexibility of our approach for the generation of compact, high-quality feature preserving meshes from various datasets including point clouds acquired by laser scanners.

Chapter 8

This last chapter summarizes the contributions of this thesis, the benefits of the proposed approaches, recalls their limitations and the possible tracks for future work.

Appendix A

This appendix describes the point set processing toolbox we developed and made publicly available through CGAL. It is particularly useful in order to filter and smooth point clouds (Chapter 6), also to orient normals attached to the points (Chapter 7).

1.4 Publications of the author

- N. Salman and M. Yvinec. *Video: High resolution surface reconstruction from overlapping multiple-views.* In Proceedings of the 25th Annual Symposium on Computational Geometry, June 2009.
- N. Salman and M. Yvinec. *Surface reconstruction from multi-view stereo.* In ACCV Workshop on Representation and Modeling of Large-Scale 3D Environments (Modeling-3D), September 2009.
- P. Alliez, L. Saboret and N. Salman. *Point set processing.* In CGAL User and Reference Manual. CGAL Editorial Board, 3.6 edition, October 2009.
- N. Salman, M. Yvinec and Q. Mérigot. *Feature preserving mesh generation from 3D point clouds.* In Proceedings of the Eurographics Symposium on Geometry Processing, Lyon, France, July 2010.

Do not worry about your
difficulties in mathematics,
I assure you that mine are greater.
Albert Einstein (1879-1955)

2 Background

In this chapter, we give some background on the basic computational geometry concepts that are needed in this thesis. We start by recalling some definitions and properties concerning the Delaunay triangulations of a set of points of $\mathbb{R}^d$, and their weighted counterparts. We then call back the notions of medial axis and ε-sampling of a surface immersed in $\mathbb{R}^3$, followed by the notion of Hausdorff distance, restricted Delaunay triangulation and its applications to surface mesh generation. For a comprehensive treatment and for references to the extensive literature on the subject one may refer to the following books [Boissonnat & Yvinec, 1998, de Berg et al., 2008] and research report [Cazals & Giesen, 2004].

2.1 Delaunay triangulations

Let $\mathcal{P} = \{p_i\}_{i=1,\dots,n}$ be a finite set of points in $\mathbb{R}^d$.

2.1.1 Voronoi diagram

The *Voronoi diagram* associated to $\mathcal{P}$, denoted $V(\mathcal{P})$, is a cellular decomposition of $\mathbb{R}^d$ into d-dimensional convex polytopes called *Voronoi cells*. There is one cell for each point p_i of $\mathcal{P}$ and the Voronoi cell of a point p_i, denoted $V(p_i)$, is composed of the set of points of $\mathbb{R}^d$ closer to p_i than to any other point in $\mathcal{P}$:

$$V(p_i) = \{p \in \mathbb{R}^d : \forall j \neq i, \|p - p_i\| \leq \|p - p_j\|\}.$$

The Voronoi cell $V(p_i)$ can also be considered as the intersection of $n-1$ half-spaces. Each such half-space contains p_i and is bounded by the *bisector planes* of segments $[p_i p_j]$, $j \neq i$. $V(p_i)$ is therefore a convex polytope, possibly unbounded.

In two dimensions, *Voronoi edges*, are the edges shared by two Voronoi cells, and *Voronoi vertices* are the points shared by three or more Voronoi cells. In three dimensions, *Voronoi facets*, *edges* and *vertices* are the geometric structures shared by two, three and four Voronoi cells respectively.

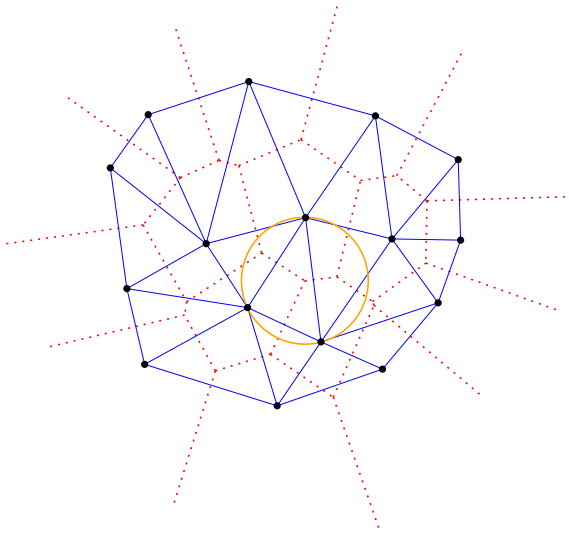


Figure 2.1: The Voronoi diagram of a set of points (red) is dual to the Delaunay triangulation (blue), which satisfies the empty circle property (orange).

2.1.2 Delaunay triangulation

The *Delaunay complex* $D(\mathcal{P})$ of $\mathcal{P}$ is the geometric dual of the Voronoi diagram $V(\mathcal{P})$ in the following sense. Whenever a collection $V(p_1), \ldots, V(p_k)$ of Voronoi cells have a non-empty intersection, the convex cell whose vertices are $p_1, \ldots, p_k$ belongs to the Delaunay complex. If we assume that the sampling $\mathcal{P}$ is in *general position*, *i.e.* includes no subset of $d+2$ cospherical points, any cell in the Delaunay complex is a simplex. This cellular complex is then a triangulation, called the *Delaunay triangulation*. It describes a partition of the convex hull of $\mathcal{P}$ into d-dimensional simplices (edges for $d=1$, triangles for $d=2$, tetrahedra for $d=3$).

Each simplex in the Delaunay triangulation is dual to a Voronoi face. In two dimensions, the dual of a Delaunay triangle is the Voronoi vertex which is also the circumcenter of the triangle. In three dimensions, each Delaunay tetrahedron is the dual of a Voronoi vertex which is the tetrahedron's circumcenter. The dual of a Delaunay facet is a Voronoi edge, the dual of a Delaunay edge is a Voronoi facet, and the dual of a Delaunay vertex p_i is a Voronoi cell $V(p_i)$. Figure 2.1 illustrates these different dual relationships in two dimensions.

The Delaunay triangulation can also be defined through the *empty circle* (respectively empty sphere in 3D) property [Delaunay, 1934]. A triangle (respectively a tetrahedron) is in the Delaunay triangulation if and only if its circumcircle (respectively its circumsphere) does not enclose any other points of $\mathcal{P}$.

This type of triangulation comes with several nice properties that explain its frequent use in geometric modeling, especially in surface reconstruction. In particular in 2D, the Delaunay triangulation maximizes the smallest of its triangles angles. Another useful property is that its external facets form the boundary of the convex hull of the set of triangulated points.

Complexity In 3D, the algorithmic complexity of the Delaunay triangulation is $O(n^2)$ or more generally $O(n\log n + n^{\lceil \frac{d}{2} \rceil})$ in d-dimensions [Boissonnat & Yvinec, 1998]. For a locally uniform ε-sampling of a surface, the number of cells in the 3D Delaunay triangulation is quasi-linear compared to n [Attali et al., 2003].

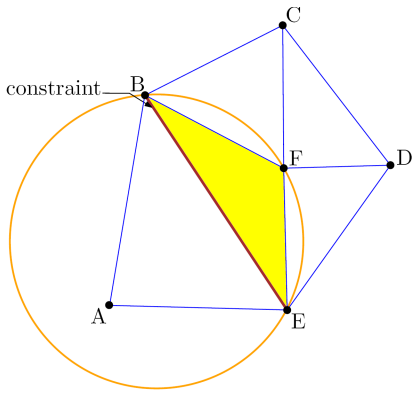


Figure 2.2: A constrained Delaunay triangulation. Point A is invisible from the yellow triangle, the triangle BFE is said constrained Delaunay.

2.1.3 Constrained Delaunay triangulation

The definition of a two dimensional *constrained Delaunay triangulation* is based on a notion of visibility. Let $\mathcal{G}$ be a set of constraints forming a straight-line planar graph. Two points p and q are said to be visible from each other if the interior of segment $[pq]$ does not meet any segment of $\mathcal{G}$. A *constrained Delaunay edge* is an edge e such that there exists a circle through the endpoints of e that encloses no vertices of $\mathcal{T}$ visible from an interior point of e (see Figure 2.2). The triangulation $\mathcal{T}$ of a set of points in $\mathbb{R}^2$ is a *constrained Delaunay triangulation* of $\mathcal{G}$, if each edge e of $\mathcal{T}$ is either an edge of $\mathcal{G}$ or a constrained Delaunay edge.

2.2 Power diagram and regular triangulation

Here we take as our finite set of objects a set of balls (instead of points) and consider as distance function of a point p to a ball $B(c, r)$ of radius r centered at c in $\mathbb{R}^d$, the power of p to B.

2.2.1 Power diagram

We call *power* of a point p to a ball $B(c, r)$ the real number

$$d_{\text{pow}}(p, B) = \|p - c\|^2 - r^2 \,.$$

Let $B_1(c_1, r_1)$ and $B_2(c_2, r_2)$ two balls in $\mathbb{R}^d$. The *power distance* between B_1 and B_2 is defined as follows:

$$d_{\text{pow}}(B_1, B_2) = \|c_1 - c_2\|^2 - r_1^2 - r_2^2 \,.$$

By using this distance we can generalize the Voronoi diagram of a set of points to a set of balls $\mathcal{B} = \{B_1, \ldots, B_n\}$ such that the *power cell* $P(B_i)$ of a ball B_i is the region of space where a non-weighted point p is closer to B_i, in terms of power distance, than to any other ball in $\mathcal{B}$:

$$P(B_i) = \{p \in \mathbb{R}^d : \forall j \neq i, d_{\text{pow}}(p, B_i) \leq d_{\text{pow}}(p, B_j)\} \,.$$

It can be easily shown that the bisector, in terms of power distance, of two balls is a hyperplane orthogonal to the line passing through their centers. This corresponds to the bisector of their

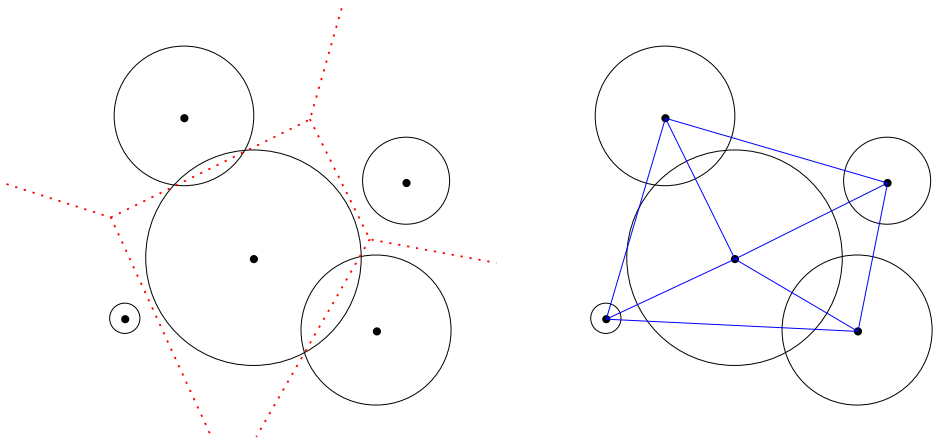


Figure 2.3: The power diagram (left) and regular triangulation (right) of a set of five circles.

centers if they have equal radii. The intersection (non-empty) of the half-spaces bounded by the bisecting hyperplanes between B_i and B_j for all $i \neq j$ and that contains the points of space with smaller power distance to B_i than to B_j, is a convex polytope.

The different power cells of $\mathcal{B}$ induce the *power diagram*: the cell complex containing power regions and their faces. Note that similarly to the Voronoi cells, power cells may be unbounded. The power diagram of a set of balls of equal radii coincides with the Voronoi diagram of their centers. Power diagrams are very similar to Voronoi diagrams: the main difference is that some balls may have an empty power region. Figure 2.3 illustrates a two-dimensional example of the power diagram of five circles.

2.2.2 Regular triangulation

Taking the geometric dual of the power diagram yields what is called the *regular triangulation*. The duality is defined in exactly the same way as for Voronoi diagrams and Delaunay triangulations. That is why regular triangulations are also referred to as *weighted Delaunay triangulations*. Namely, the regular triangulation contains an edge between the centers of two balls if and only if their associated power cells share a common facet. As for points, we obtain a triangulation of the centers of the balls in $\mathcal{B}$, *i.e.* a partition of their convex hull into simplices of dimension d. Similarly to the power diagram, the regular triangulation of a set of balls of equal radii corresponds to the Delaunay triangulation of their centers. Figure 2.3 illustrates a two dimensional example of the regular triangulation of five circles.

Two balls are orthogonal if their power distance is zero. In $\mathbb{R}^d$, $d+1$ balls have a unique common orthogonal ball called the *power ball*. It can be straightforwardly verified that this ball coincides with the circumsphere of their centers if they are all zero weighted points. The empty sphere property of the Delaunay triangulation extends to the case of power distance. A triangulation of a set of balls is regular if the power balls of all simplices are *regular*, *i.e.* their power distances to all the balls is non-negative.

A ball can intersect or not its associated power cell, it can also have no power region in the diagram, in which case no power cell is associated to it. A ball with no associated power cell does not appear in the regular triangulation.

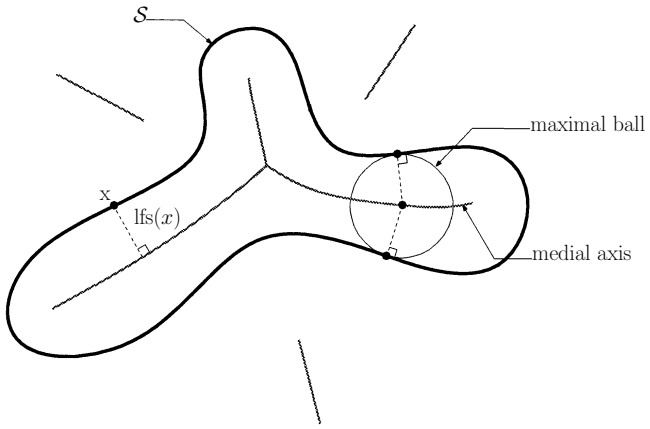


Figure 2.4: Medial axis of a planar curve $\mathcal{S}$.

2.3 Medial axis and ε-sampling

Let $\mathcal{S}$ be a smooth surface immersed in $\mathbb{R}^3$ and $\mathcal{P} \subset \mathcal{S}$ a finite sample of points on the surface $\mathcal{S}$.

2.3.1 Medial axis

A ball in $\mathbb{R}^3$ is said to be *empty* if it does not contain any point of $\mathcal{S}$ inside of it. An empty ball is *maximal* if it is not included in any other empty ball. The *medial axis* $M(\mathcal{S})$ of $\mathcal{S}$ is the set of all centers of maximal empty balls. The maximal empty balls are tangent to $\mathcal{S}$ in two or more points (exception at the curvature maximum) of $\mathcal{S}$. Figure 2.4 illustrates the medial axis of a planar domain.

The transformation that associates to a shape its medial axis is not continuous. A small perturbation of the shape can lead to a big perturbation of the medial axis. Thus, it is hard to compute the medial axis. We can however get a stable approximation of the medial axis of a surface from the Voronoi diagram of a sample of points on the surface [Attali et al., 2009].

2.3.2 Local feature size

The *local feature size* at a point $x \in \mathcal{S}$, noted lfs(x), is a function lfs : $\mathcal{S} \to \mathbb{R}$ that measures the minimal distance of x to the medial axis $M(\mathcal{S})$ of $\mathcal{S}$:

$$\text{lfs}(x) = \min_{y \in M(\mathcal{S})} \|x - y\| \ .$$

In their work [Amenta & Bern, 1998] proved that the local feature size is a 1-Lipschitz function, *i.e.* $\text{lfs}(x) \leq \text{lfs}(y) + \|x - y\|$ for all $x, y \in \mathcal{S}$.

The function lfs can be seen as a measure of both the thickness and the local curvature of an object. Ambiguities arise in reconstruction processes as soon as the samples are not dense enough with respect to the local feature size of the shape.

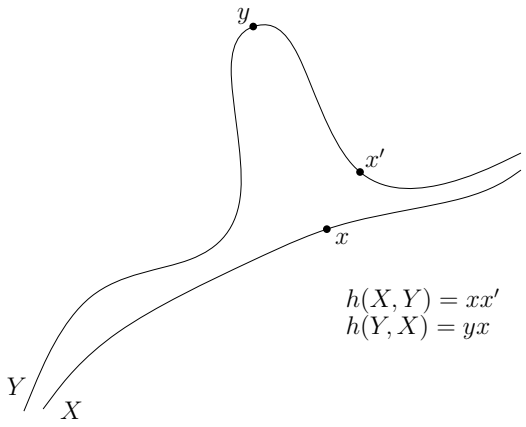


Figure 2.5: The one-sided Hausdorff distance is asymmetric.

2.3.3 Sampling conditions

The sampling conditions allow to deduce properties of the surface from the point sample, and in particular to estimate normals.

ε-sample A point sample $\mathcal{P}$ is called ε-sample of a surface $\mathcal{S}$ if and only if for each point $x \in \mathcal{S}$, there exist a point $p \in \mathcal{P}$ at distance at most $\varepsilon \text{lfs}(x)$, *i.e.* $\|x - p\| \leq \varepsilon \text{lfs}(x)$.

The ε-sample criterion guarantees that a surface $\mathcal{S}$ is densely sampled in the regions of high curvature and where two separate layers of the surface get close to each other. On one hand [Dey et al., 2003] and on the other [Funke & Ramos, 2002] introduced the more general notion of *locally uniform* ε-sample.

(ε, δ)-sample A point sample $\mathcal{P}$ is called (ε, δ)-sample of a surface $\mathcal{S}$ for $\delta < \varepsilon \leq 1$ if $\mathcal{P}$ is an ε-sample of $\mathcal{S}$ and $\|p - q\| \geq \delta \text{lfs}(p)$ for all $p, q \in \mathcal{P}$.

The ε-sample criterion gives a lower bound over the density of the sample, while the criterion based on δ gives an upper bound and controls the position of the sample points to ensure a certain uniformity of the sampling.

2.4 Hausdorff distance

The *Hausdorff distance* is a measure of the distance of two subsets of some metric space. In this thesis we are interested in the case where these subsets are surfaces in $\mathbb{R}^3$.

Given two closed subsets X, Y of $\mathbb{R}^3$, the *one-sided Hausdorff distance* denoted $h(X, Y)$ is defined as:

$$h(X, Y) = \max_{x \in X} \min_{y \in Y} \|x - y\| \ .$$

The one-sided Hausdorff distance, not as the name might suggest, is not a distance measure since in general it is not symmetric. Symmetrizing h yields the Haussdorff distance as $H(X, Y) = \max(h(X, Y), h(Y, X))$ (see Figure 2.5).

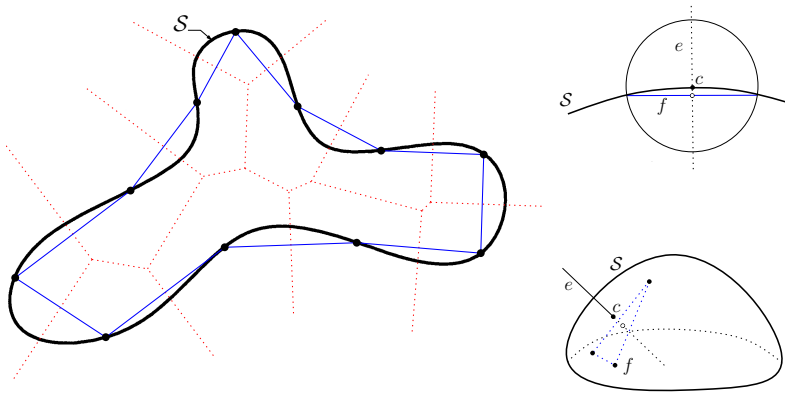


Figure 2.6: Left: the Voronoi diagram (red) and the Delaunay triangulation edges (blue) of a point set restricted to a planar closed curve $\mathcal{S}$. Right: on the top, the equivalent of a surface Delaunay ball for a planar closed curve $\mathcal{S}$; on the bottom, the center of a surface Delaunay ball of a facet $f \in D_{\mathcal{S}}(\mathcal{P})$ is the intersection of the Voronoi edge e, dual of f, with the surface $\mathcal{S}$.

2.5 Restricted Delaunay triangulation and mesh generation

The notion of restricted Delaunay triangulation to a surface was first introduced by [Chew, 1993] through a surface sampling algorithm, and then formalized by [Edelsbrunner & Shah, 1997], who studied its properties. In particular they proved that the restricted Delaunay triangulation is homeomorphic to the surface under some sampling conditions.

We call *restricted Delaunay triangulation* of $\mathcal{P}$ to $\mathcal{S}$, denoted by $D_{\mathcal{S}}(\mathcal{P})$, the sub-complex of the Delaunay triangulation $D(\mathcal{P})$ formed by the so-called *restricted* simplices. A simplex is said to be *restricted* to $\mathcal{S}$ if its dual Voronoi simplex intersects $\mathcal{S}$. In three dimensions, the Delaunay triangulation restricted to a surface is formed of the Delaunay faces whose dual Voronoi edges intersect the surface. Left part of Figure 2.6 illustrates an example in two dimensions.

Let f be a facet of $D_{\mathcal{S}}(\mathcal{P})$ and let e be its dual Voronoi edge, any point of $e \cap S$ is the center of a circumball of f empty of points of $\mathcal{P}$. Hereafter, such a ball is called a *surface Delaunay ball* of P (see right part of Figure 2.6).

Edelsbrunner and Shah proved that $D_{\mathcal{S}}(\mathcal{P})$ is homeomorphic to $\mathcal{S}$ if each Voronoi cell satisfies the *closed ball property*, that is if every Voronoi cell is homeomorphic to a closed ball. In their work, [Amenta et al., 2000] proved that if $\mathcal{P}$ is an ε-sample, with $\varepsilon \leq 0.06$, then the restricted Delaunay triangulation $D_{\mathcal{S}}(\mathcal{P})$ is a piecewise smooth linear surface homeomorphic to $\mathcal{S}$. From a geometrical point of view, the Hausdorff distance between $D_{\mathcal{S}}(\mathcal{P})$ and $\mathcal{S}$ can be made arbitrarily small, and normals and curvature can be consistently approximated from the restricted Delaunay triangulation.

Building on these results, the meshing algorithm of [Boissonnat & Oudot, 2005] maintains the restricted Delaunay triangulation $D_{\mathcal{S}}(\mathcal{P})$ of a sampling $\mathcal{P}$ of the surface, and refines the sampling $\mathcal{P}$ until it is dense enough for the restricted Delaunay triangulation to be homeomorphic to the surface $\mathcal{S}$ and to be a good approximation of $\mathcal{S}$ in the Hausdorff sense. For a more detailed description of this algorithm please refer to Section 6.2.1, and to Chapter 5 for a broad view on the main automatic simplicial mesh generation techniques.

When you can measure
what you are talking about
and express it in numbers,
you know something about it.
William T. Kelvin (1824-1907)

3

Acquisition of data sets

Different technical solutions have been developed to acquire 3D coordinates of points on an object surface. In this chapter we will briefly review some of the principal solutions in order to help the reader understand the data sets used to evaluate our reconstruction algorithms. For more details we refer the reader to the course notes [Curless & Seitz, 2000] and PhD thesis [Labatut, 2009].

First, a quick overview of laser range scanning techniques is presented, then two different approaches to compute point clouds from supposedly calibrated images are provided. Finally, we will present the novel idea behind the Gyroviz project whose main motivation stems from the observation of the limitations of the previous stereo approaches. Also, the availability of lines of sight information associated to the point clouds acquired by means of multi-view passive stereo is underlined. As will be seen in Chapter 6, this information will be of major importance in our visibility based smooth surface reconstruction framework.

3.1 Point sets from laser scanners

When we want to capture an arbitrary surface, we need to sample 3D points densely across the sequence of measurements. This consists in measuring the distance between the capturing device and the target object. There exists a large number of optical scanners, based on different techniques and adapted to different uses, from terrain survey and mapping to microscopic studies. Here, we will talk about the two most common optical methods: time-of-flight and triangulation scanning principles.

The **Time-of-Flight** (ToF) systems shown in Figure 3.1 work on the principle of return-time of an emitted laser pulse. Namely, a laser pulse is emitted to the object surface and the distance between the emitter and the reflecting surface is computed from the travel-time between pulse emission and reception. These systems use rotating devices for the angular deflection of the laser pulse and generally use simple algorithms for distance computation which may result in poor accuracy. Typically, the **error** of the range measurements by time-of-flight systems are within millimeters (e.g. 2mm for the Photon 120 scanner, 6mm for the ScanStation C10 scanner, 7mm for the Optech ILRIS-3D, 10mm for the I-SiTE 8800 scanner). However, this accuracy remains stable throughout the whole object surface as distances remain relatively short.

These systems **range** can reach several hundred meters and are often used in big-scale applications like scanning of terrains in different environments. In practice, the range is limited by the

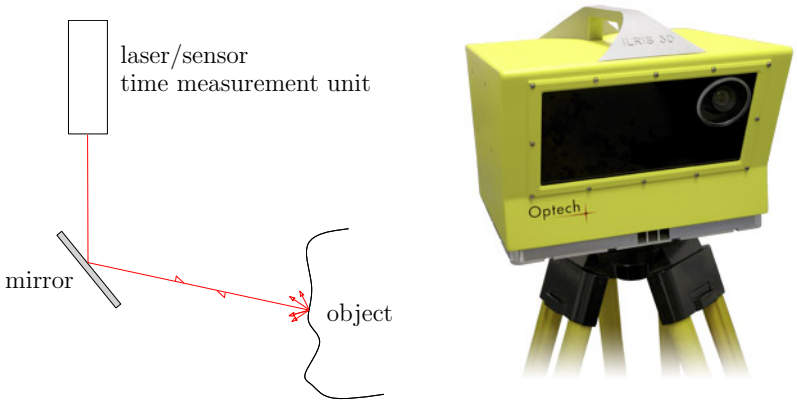


Figure 3.1: Time-of-flight scanner principle (left); Optech^{TM}ILRIS-3D time-of-flight LIDAR scanner (right).

Note: One can practically attain results that challenge the maximum accuracy declared by the ToF scanners manufacturers through a careful 3D acquisition and processing procedure [Callieri et al., 2009].

requirement that the system does not harm pedestrians in the targeted area and by the gradual spread of the beam with distance. Of more interest in this thesis, the use of time-of-flight optical remote sensing technology (LIDAR) is growing rapidly, both in terms of the number of application domains and in the prevalence of the method in real-world practice.

The second setup shown in Figure 3.2 for **triangulation systems** consists of an emitting device, sending a laser beam at a defined, incrementally modified angle from one end of a mechanical base onto the target object, and a CCD array (sensor) at the other end of this base which records the laser beam (or stripe) positions on the object. Since the base distance between the emitter and the sensor is known, as well as the orientation of the sensor and the direction of the emitter, depth can be measured and the 3D position of the reflecting surface element can be inferred.

The **accuracy** of triangle-based systems decreases with the square of the distance from the emitting device to the object to measure [Everett, 1995]. Accuracy is improved however with greater base distances between the emitter and the sensor but obviously this can also result in "occluded" areas, to which either the emitter or the sensor loses the line of sight (see Figure 3.3). A thorough analysis of the different source of measurement errors in triangulation range scanners can be found in [Curless, 1997]. Accuracy therefore depends, *inter alia*, on the range for which the scanner will be used. The absolute accuracy for such scanners is typically counted in a few hundredths of a millimeter but their **range** is short, varying from a few centimeters to a few meters, thus are mainly restricted to small-scale objects.

A few numbers to give orders of magnitude, the triangulation scanning device used to acquire the dense multi-view stereo benchmark of [Seitz et al., 2006], a Cyberware^{TM}Model 15, has a depth accuracy of about 0.05mm to 0.2mm with a resolution of 0.25mm and one of the scanners used in the Aim@Shape 3D scanning repository [aas, 2006], a Minolta^{TM}Vivid, has similar specifications (see Figure 3.4 for a small object indoor scan from the Aim@Shape repository). In contrast, the accuracy of the LIDAR scanner used in the dense multi-view stereo benchmark of [Strecha et al., 2008] can be of several millimeters. The size of the scanned object or scene

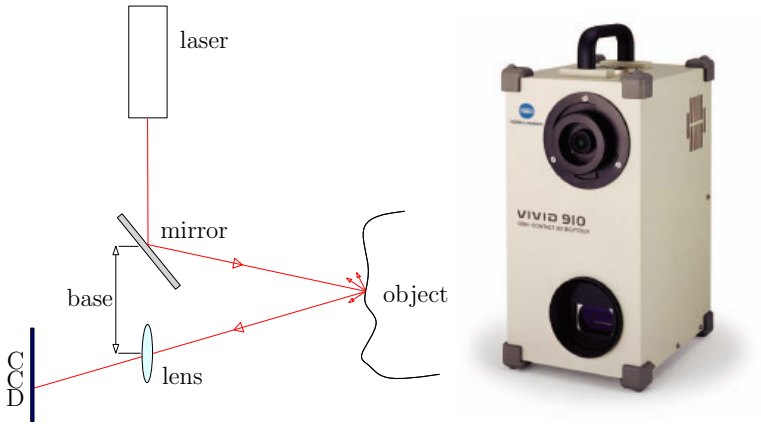


Figure 3.2: Triangulation scanner principle (left); Minolta^{TM}Vivid 910 triangulation scanner (right).

is however significantly different: a few centimeters for the triangulation scanners, and several decimeters for the LIDAR scanner (see Figure 3.4 for an example of a large-scale outdoor scan).

Note: Both time-of-flight and triangulation range finders are susceptible to produce a variety of incorrect points in the vicinity of edges. This is due to a weaker reflected signal, or because the laser pulse hits the front and the back of the edge simultaneously. The coordinate for a point that has hit the edge of an object will be computed based on an average, therefore giving a noisy response for the depth value in that area. Typically, a range error may vary in this case from just a tenth of a millimeter to values of several decimeters [Boehler et al., 2003]. This is something to keep in mind while reading Chapter 7.

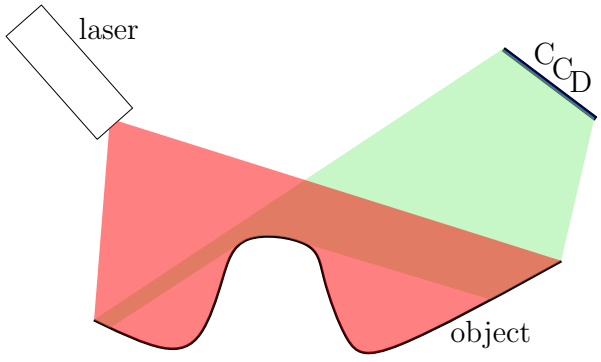


Figure 3.3: One of the acquisition problems: "occluded" areas. The visibility of the emitter (red) and the CCD receiver (green). Some surface regions may be visible from the emitter and not visible from the receiver and vice versa.

While technological advances on laser range scanning devices have greatly increased the availability of very high quality geometric measurements, in terms of density, uniformity and accuracy of sampling, some clear drawbacks raise when compared side-by-side with passive multi-view stereo approaches:

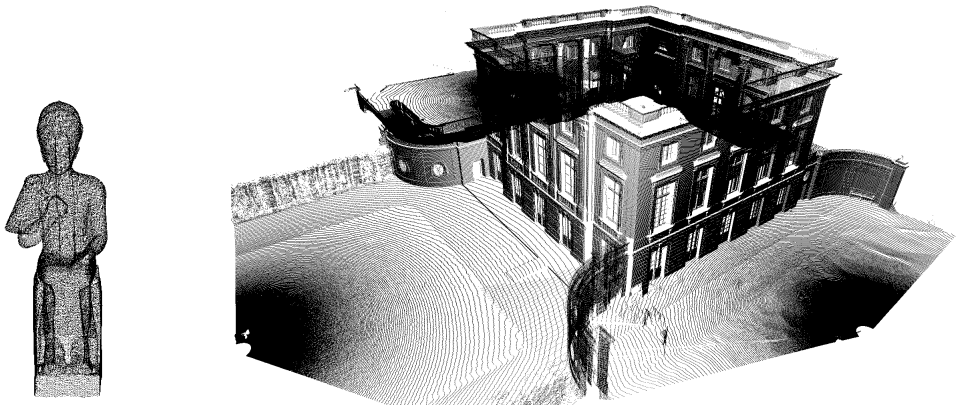


Figure 3.4: Point cloud from a triangulation scanner source Aim@Shape *Ramses* (left); from a LIDAR scanner source *Trianon* (*courtesy of Livio De Luca*)(right).

- financial cost of scanning devices is not to be underestimated $70k-250k. It is way more expensive than a standard consumer high-definition camera used by multi-view passive stereo approaches;
- the acquisition process can be time consuming and necessitate laborious setups especially for the acquisition of large-scale scenes;
- the number of generated points is huge: data acquisition rate can be quite high, reaching more than half a million points a second for some laser scanner models (Faro Photon 120) thus, the acquisition of billion sample points becomes more and more familiar. Such data sets demand dedicated algorithms to either first downsample them [Wu & Kobbelt, 2004] to make the surface reconstruction problem more tractable or require sophisticated out-of-core [Bernardini et al., 1999] or streaming reconstruction algorithms [Bolitho et al., 2007] to handle this huge amount of data with a reasonable memory footprint.

3.2 Point sets from multi-view passive stereo

A relevant alternative to point-sample an object or a scene is to use passive stereo techniques. Opposed to laser scanners, high definition digital cameras are now common devices disseminated in the mass market. However, to relate corresponding points in different images one cannot simply rely on the acquisition of many images from various positions.

Considering only the captured images, one principal problem consists in finding the *structure from motion*, where both the scene shape (*structure*) and the camera parameters (*motion*) consistent with a set of correspondences between the scene and image features are estimated [Pollefeys et al., 2004]. In this process, the intrinsic camera parameters (focal length, coordinates of the principal point, *etc*) are often supposed to be known [Nistér, 2004], or recovered *a posteriori* through auto-calibration [Triggs et al., 2000, Pollefeys et al., 2004]. A final *bundle-adjustment* is then typically used to optimize both camera parameters and 3D points positions of the tracked features with a large non-linear least-squares optimization [Triggs et al., 2000, Lourakis & Argyros, 2009]. An exhaustive overview of the methods for estimating the epipolar geometry can be found in the computer vision reference books of [Faugeras et al., 2001, Hartley & Zisserman, 2004].

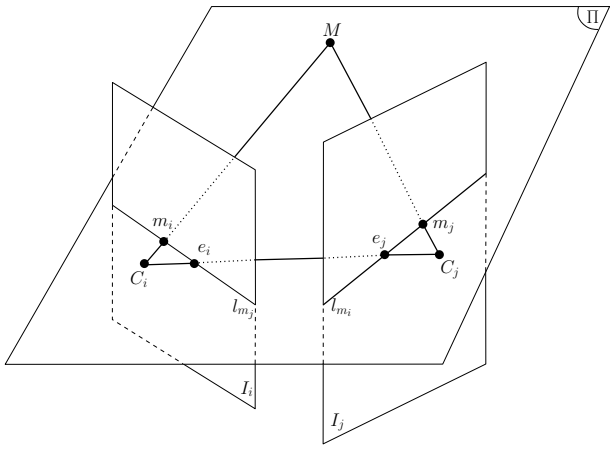


Figure 3.5: The epipolar geometry of two cameras indicated by their centers C_i and C_j.

In the following, we assume that n calibrated views are given, each with a corresponding image I_i and a camera P_i that represents the mapping from the 3D point $M = [x, y, z]^T$ in a world coordinate to its image coordinates $m_i = [u, v]^T$ in I_i through the camera center C_i. We also assume that images have been corrected for radial distortion and that the cameras are modeled with the widely used pinhole model. Both world coordinates and image coordinates are related by

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P_i \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} ,$$

where P_i is a a 3x4 matrix, called the *perspective projection matrix*. Also, we denote the homogeneous coordinates of a vector $v = [x, y, \ldots]^T$ by $\tilde{v} = [x, y, \ldots, 1]^T$, we have

$$\tilde{m}_i = P_i \tilde{M} .$$

The matrix P_i can be decomposed as $P_i = A[R\ t]$ where A is a 3 x 3 matrix, called the *camera intrinsic matrix*, that maps the normalized image coordinates to the original image coordinates, and $[R\ t]$, called the *extrinsic parameters*, the 3D rotation and translation displacement from the world coordinate system to the camera coordinate system.

Given a point m_i in the image I_i, its corresponding point m_j in the image I_j is constrained to lie on a line called the *epipolar line* of m_i, denoted by l_{m_i} (see Figure 3.5). The line l_{m_i} is the intersection of the *epipolar plane* Π, defined by m_i, and the camera centers C_i, C_j, with the image plane I_j. Furthermore, all epipolar lines of the points in the image I_i pass through a common point e_j called the *epipole* and defined as the intersection of the line C_iC_j with the image plane I_j.

In order to match m_i and m_j, the following equation must be satisfied:

$$\tilde{m}_j^T F_{ij} \tilde{m}_i = 0$$

where F_{ij} is called the *fundamental matrix* [Luong & Faugeras, 1996] of the oriented camera pair (i, j). Algebraically, $F_{ij}\tilde{m}_i$ defines the epipolar line of l_{m_i} in image I_j. Intuitively, this equation says that the correspondence in I_j of point m_i lies on the epipolar line l_{m_i}.

Assuming that the choice of camera pairs is given, in the following we will briefly present two standard point-cloud generation techniques from passive stereo. Namely, identifying "interest

points" in the images and then matching them across the images, or by plane sweeping for sparse depth maps computation. The two techniques we will describe and use are borrowed from [Vu et al., 2009]. In order to test our surface fitting framework in Chapter 6 we will mainly rely on the point clouds generated by the second approach, favoring density over matching robustness.

3.2.1 Interest points

One common approach to acquire a 3D point cloud from a set of calibrated images is to start by identifying interest points throughout all the images. To capture most of the geometry of the sampled shape, [Vu et al., 2009] use two complementary types of interest points: Harris corners which are typically lying on "corners" in images and Difference-of-Gaussian located at the center of "blob-like" structures in images. In what follows we will briefly recall the definitions and origins of these standard interest points.

Multi-scale Harris corners Given a grayscale two dimensional image I, this image is convolved with a standard two dimensional Gaussian kernel with width σ

$$G^\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} ,$$

to give a *scale-space* representation [Lindeberg, 1993], $I^\sigma = I * G^\sigma$, used to detect interest points at *local scale* σ.

We now consider taking the shifted version K^σ of the image I^σ by a vector (u,v):

$$K^\sigma(x,y) = I^\sigma(x+u, y+v) .$$

Let $I_x^\sigma = \partial_x I^\sigma$ and $I_y^\sigma = \partial_y I^\sigma$ denote the partial derivatives of I^σ. Moreover, let G^τ denote the weighted Gaussian window function with an *integration scale* τ, introduced to integrate over a weighted neighborhood of a point. The shifted image K^σ is approximated by a Taylor expansion truncated to the first order terms:

$$K^\sigma(x,y) \approx I^\sigma(x,y) + [I_x^\sigma(x,y)\, I_y^\sigma(x,y)] \begin{bmatrix} u \\ v \end{bmatrix} .$$

The corresponding weighted sum of squared differences $S^{\sigma,\tau}$ between both images I^σ and K^σ can then be approximated by:

$$S^{\sigma,\tau}(u,v) \approx (u,v) M^{\sigma,\tau} \begin{pmatrix} u \\ v \end{pmatrix} ,$$

where $M^{\sigma,\tau}$ is called the *multi-scale second moment matrix* [Lindeberg & Gårding, 1997] that captures the intensity structure of the local neighborhood and is defined as:

$$M^{\sigma,\tau} = G^\tau * \begin{pmatrix} {I_x^\sigma}^2 & I_x^\sigma I_y^\sigma \\ I_x^\sigma I_y^\sigma & {I_y^\sigma}^2 \end{pmatrix} .$$

Matrix $S^{\sigma,\tau}$ captures the intensity structure of the local neighborhood, thus large values of $M^{\sigma,\tau}$ characterize corners at an image scale σ. If λ_1, λ_2 are the eigenvalues of matrix $M^{\sigma,\tau}$, then there are three cases to be considered:

- If both λ_1 and λ_2 are small, so that local function $S^{\sigma,\tau}(x,y)$ is flat, the neighborhood of a point is of approximately constant intensity, thus not an interest point.

- If one eigenvalue is high and the other low, then only local shifts in one direction cause little change in $S^{\sigma,\tau}(x,y)$ and significant change in the orthogonal direction; this indicates an *edge*.
- If both eigenvalues are high, then shifts in any direction will result in a significant increase; this indicates a *corner*.

Instead of computing explicitly the eigenvalues of $M^{\sigma,\tau}$ to find corners, strong local maxima of the Harris measure [Harris & Stephens, 1988] applied to $M^{\sigma,\tau}$, denoted $harr(M^{\sigma,\tau})$ are used:

$$harr(M^{\sigma,\tau}) = \det(M^{\sigma,\tau}) - \kappa \mathrm{Tr}(M^{\sigma,\tau})^2 = \lambda_1\lambda_2 - \kappa(\lambda_1+\lambda_2)^2\,,$$

where κ is a sensitivity parameter that has to be determined empirically. In the literature, values in the range of 0.04 to 0.06 have been reported practicable. Also, this measure is usually referred to as the *multi-scale Harris corner measure*.

Difference-of-Gaussian In the computer vision field, "blob detection" refers to the modules that aim at detecting points or regions that are either darker or brighter than the surrounding. One of the most common blob detectors is based on the Laplacian of the Gaussian (LoG). Given an input image I, apply the Laplacian operator Δ to the scale-space representation I^σ of I:

$$\Delta I^\sigma = I^\sigma_{xx} + I^\sigma_{yy} = \Delta(I * G^\sigma) = (\Delta G^\sigma) * I\,.$$

Remark that the convolution and Laplacian operations are both linear and shift invariant, thus their computation order can be interchanged.The LoG's computation results in strong positive responses for dark blobs and strong negative responses for bright blobs of size $\sqrt{\sigma}$. Using this, the position and characteristic scale of blobs can be found by detecting strong local extrema of the *scale-normalized* LoG image representation $\sigma^2\Delta I^\sigma$.

In practice, the scale-normalized LoG operator can be approximated as the limit case of the difference between two Gaussian smoothed images [Marr & Hildreth, 1980] (scale-space representations):

$$\sigma^2\Delta I^\sigma \approx I^{k\sigma} - I^\sigma\,,$$

where k is a constant multiplicative factor usually close to 1. This approach is referred to as the Difference of Gaussians (DoG) approach. In a similar fashion as for the LoG blob detector, blobs can be detected from scale-space extrema of the DoGs.

Note: Interest points, called keypoints in the SIFT framework [Lowe, 2004], are identified as local scale-space extrema of the DoG images. The SIFT descriptors are used by [Labatut et al., 2007], later revamped into [Vu et al., 2009].

3.2.2 Matching interest points

Assuming that several types of interest points have been identified (multi-scale Harris corners, DoGs), and that these image points are likely to correspond to different views of the same points in the scene. The problem is then to decide which pairs of interest points, *de facto*, correspond to the same scene 3D point.

To generate a 3D point cloud, one requires an efficient way to match them across images. All potential pairs (i,j) of images are taken and for each interest point m_i in the first image I_i of this pair, its best possible match m_j^+ is sought respecting the following epipolar constraint : m_j^+ should lie in a narrow band around the corresponding epipolar line l_{m_i} in the second image

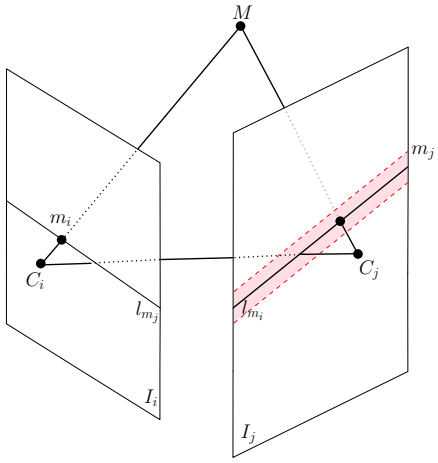


Figure 3.6: A potential match for a point can be found in a small band around the epipolar line.

plane I_j. This situation is illustrated in Figure 3.6; the width of this band is fixed and depends on the precision of the calibration process.

To find the best match m_j^+ for m_i, the image region of I_j inside a window around the potential match m_j is reprojected in the reference image I_i via a plane parallel to the focal plane of the camera P_i and passing through the potential reconstructed 3D point M. Here, it is assumed that the scene surface is locally fronto-parallel to the camera P_i, *i.e.* the optical axis is aligned with the Z-axis of the scene.

Note: Flat fronto-parallel scene surfaces are assumed in most area-based approaches. Ideally, the matching window should take the distortion into account too. But, this is a "Who comes first problem": depth is needed for that, but depth is actually what is searched for.

The matching score can then be estimated using the *Normalized Cross Correlations* (NCC) photo-consistency measure. Let W be a neighborhood around the image pixels:

$$< I_i, m_i, I_j, m_j >= \frac{1}{|W|} \sum_{w \in W} \left[(I_i(m_i + w) - \bar{I}_i(m_i)) \times (I_j(m_j + w) - \bar{I}_j(m_j)) \right] ,$$

with:

$$\bar{I}_i(m_i) = \frac{1}{|W|} \sum_{w \in W} I_i(m_i + w) ,$$

then the NCC can be defined as follows:

$$\mathrm{NCC}(I_i, m_i, I_j, m_j) = \frac{< I_i, m_i, I_j, m_j >}{\sqrt{< I_i, m_i, I_i, m_i >< I_j, m_j, I_j, m_j >}} .$$

Since the choice of an appropriate neighborhood W size is difficult, Vu *et al.* use a *multi-level matching strategy*, where the matching criterion is the sum of Normalized Cross Correlations for multiple fixed window sizes [Labatut, 2009].

The best matching interest point m_j^+ is then validated only if its matching score is above some threshold and if it is also respectively validated the other way round, meaning that the reference point has to be the best matching interest point of its best matching interest point.

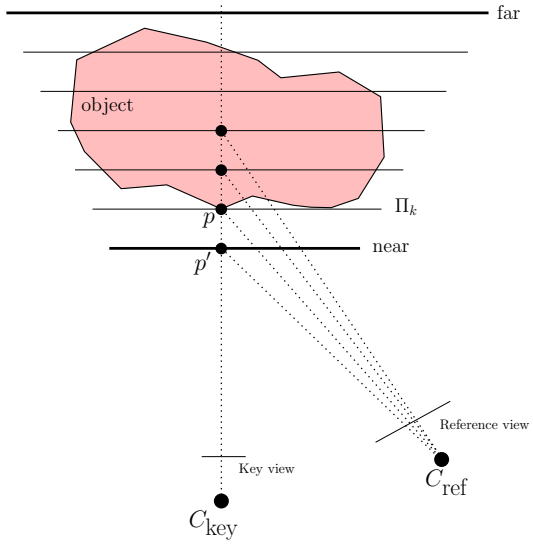


Figure 3.7: Plane sweeping principle. A visible point of the object lying on one of the planes Π_k at a point p, this point will be seen by both cameras with the same color. A point p' lying on a plane but not on the surface of the object, will not be seen by the cameras C_{key} and C_{ref} with the same color.

It is only when both conditions are satisfied that an initial 3D point can be reconstructed from the camera calibration and using standard triangular optimization [Hartley & Zisserman, 2004] by minimizing the 3D point's squared euclidean distance to the rays.

Finally, in order to take into account all possible points in other images (other than the initial pair of images) which could match with the reference point, the initial 3D point is projected in the other views, other potential unmatched interest points whose distance is less than a given threshold are located using a 2D Delaunay triangulation data structure (see Section 2.1.2 for background on Delaunay triangulations). The closest unmatched interest points are then merged with the original pair and a new 3D point is re-estimated from all the interest points.

The result of the following interest point matching step is a *set of 3D points remembering what images have observed them*, with potentially high amounts of noise and outliers as it is illustrated on the left of Figure 3.8 .

3.2.3 Sparse depth maps

When regions with large areas of insufficient textures are observed in the scene, the aforementioned multi-view stereo approach, *i.e.* extracting and matching interest points, tends to generate scattered points. This is particularly true in the case of large-scale architectural outdoor scenes, where strong planar structures occur in abundance.

In their work, [Vu et al., 2009] start by computing a set of sparse depth maps between pairs of images using a GPU-accelerated *plane sweep* approach [Collins, 1996, Yang et al., 2002]. The 3D space is iteratively traversed by a discrete number of equally spaced parallel planes between the near far planes of a particular key view, as depicted in Figure 3.7. If the plane at some depth passes through the surface of the object, the colors from the back-projected key image and from the mapped reference images should be similar assuming Lambertian surfaces, *i.e.*

surfaces that reflect light equally in all directions; conversely the colors from the back-projected key image and from the mapped reference images will differ significantly where the surface of the object does not intersect the plane. Vu *et al.* use the same thresholded multi-level NCC measure as in Section 3.2.2 to assign the best matching depth value to the pixels of the key view. By sweeping this plane through the 3D space between the near-far planes of the key view, a correlation volume is generated. Applying a winner-takes-all strategy in this volume along the rays of the key view gives then a depth map for that key view.

These initial depth maps are then fused and clusters of points are formed according to their position in different camera frustums. An initial point set is then created by hierarchically dividing the point clusters until the bounding boxes of their projections in the images are small enough. This initial point set is then inserted into a 3D k-D tree [Bentley, 1975] data structure. The latter being used to find the k nearest neighbors of each point. A plane is then fitted to each point's neighborhood using least-squares. The point is retained if the fit is good and its position is iteratively refined using the aforementioned multi-level NCC matching score. The final result is also a set of points associated to a set of cameras that have observed them. This method generates noisy point clouds with outliers. However, it yields better results than the aforementioned "extract and match" method on architectural scenes (see Figure 3.8 right).

While multi-view passive stereo methods start to be seen as a good alternative to laser scanning devices by providing some very impressive "Yes, we can" results [Agarwal et al., 2009, Furukawa et al., 2010, Frahm et al., 2010], these methods are nevertheless prone to errors due to the quasi inevitable noise in the images (known also as optical and electronic noise). When the images are noisy, the matching step in the point-sample reconstruction process becomes the critical step. The main reason is due to the multiple possible correspondences when dealing with a large number of images and interest points. Apart from noise in the images, mismatches are also, algorithmically, almost inevitable leading to numerous outliers in the resulting 3D point set. Depending on the geometry of the cameras and the repetitiveness of texture patterns, these mismatches may even aggregate in structured groups of outliers producing non-existing structures in the point cloud. Another limitation of passive stereo is the non uniform distribution of the sampling depending on the of textures in the scene.

The central motivation behind the Gyroviz project proposal, briefly described in the next section, stems from the observation of the current limitations of image-based modeling systems, which require practically a substantial amount of user interaction for the process of matching interest points between the images. This limitation becomes even more critical when dealing with massive image datasets. While a considerable amount of efforts has been put at solving this issue from the algorithmic point of view [Goesele et al., 2007, Jancosek et al., 2009], the Gyroviz project propose to address it primarily from the hardware side similar to [Pollefeys et al., 2008].

3.3 The Gyroviz project

As previously presented, for some content, automatic multi-view stereo matching procedure is inherently an ill-posed problem, when too many degrees of freedom exist (*e.g.* repeated patterns). The robustness of the process can be increased by resorting to human interaction in order to identify and to match specific features observed in multiple images of the same scene.

The Gyroviz project intends to overcome these bottlenecks by proposing an innovative approach for real time tracking from located frames. The idea is to couple an image acquisition device with a set of inertial sensors in order to obtain an accurate measure of its physical location and pose (see Figure 3.9). Both robustness and efficiency of the image matching problem are

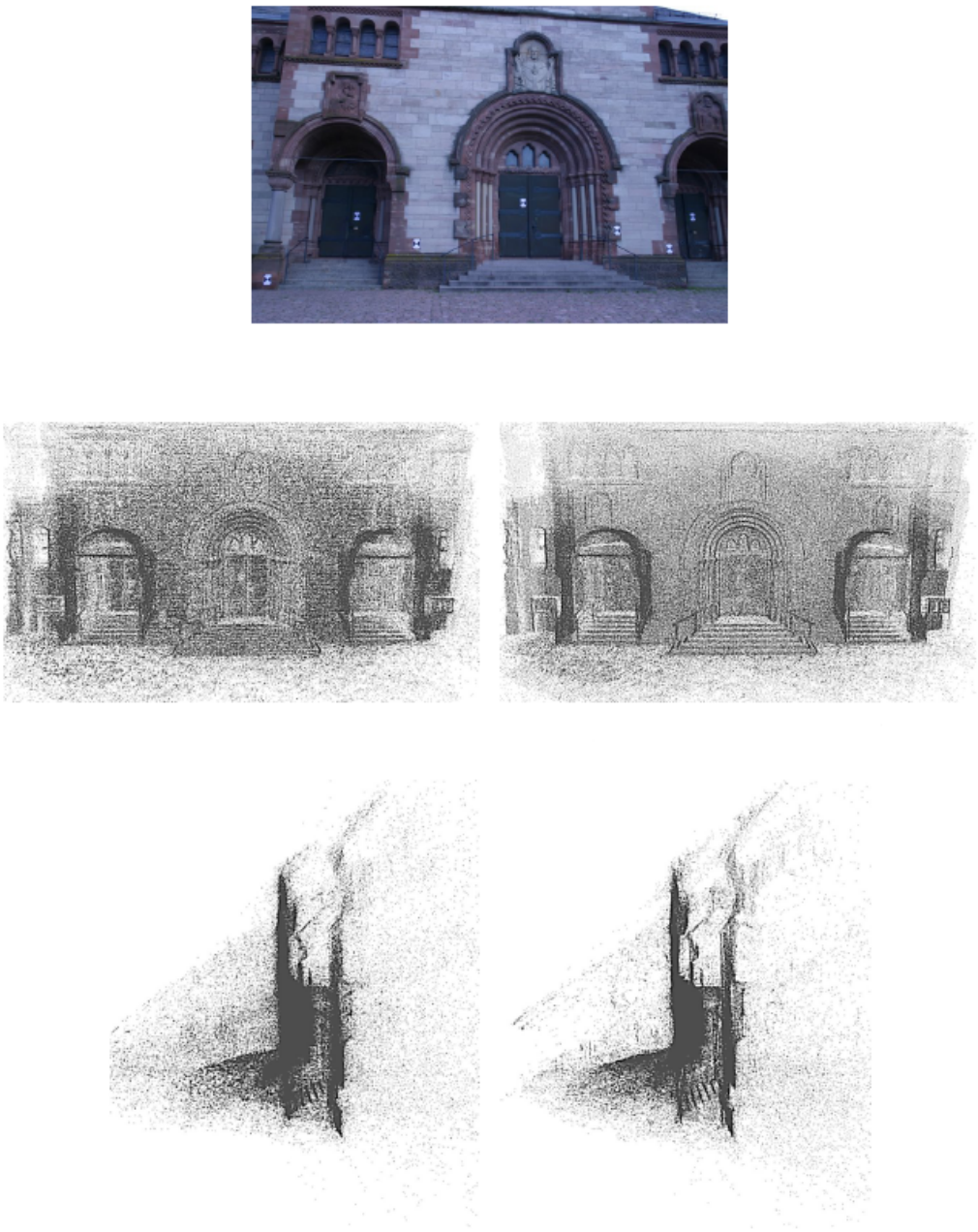


Figure 3.8: Point sets from multi-view passive stereo [Labatut, 2009]. Top: One of the *Herz-Jesu-p25* dataset [Strecha et al., 2008]. Left: Two different views of the point cloud generated using interest point extraction and matching. Right: Two different views of the point cloud generated using sparse depth maps (*courtesy of Patrick Labatut*).

Figure 3.9: The Gyroviz project: Coupling an image acquisition device with a set of inertial sensors. Left: Video camera coupled to a FOG, GPS and Guppy^{TM}camera. Right: Still camera coupled to a FOG and Guppy^{TM}camera.

this way substantially improved, as the solution space of the matching problem is drastically reduced, and the matching should always be provided with good initial guesses.

To support this scheme, the goal of the project is to include:

- a new specific inertial sensors based on "Fiber Optic Gyrometers" (FOG) which are at least a thousand times more accurate than "Micro-Electro-Mechanical" (MEM) systems and associated localization algorithms including real time inertial sensors drift compensation (please refer to [Merlo et al., 2002] if interested in more details on the FOG principles);
- a new portable acquisition system able to provide localization and shooting parameters metadata associated to each frame. By localization we mean not only the geo-localization of the acquisition device given by a low precision GPS device, but the very precise location and pose of the device through six degrees of freedom including orientation.

Part I

From 3D point clouds to meshes: State of the Art

One geometry cannot
be more true than another;
it can only be more convenient.
Henri Poincaré (1854-1912)

4

Surface reconstruction from 3D points

Given only a finite set of unorganized points $\mathcal{P} = \{p_i\}_{i=1,\dots,n}$ assumed to lie on or near an unknown surface S, the aim of surface reconstruction is to compute a surface $\mathcal{S}'$ from $\mathcal{P}$ that approximates $\mathcal{S}$.

Surface reconstruction occurs in a large number of applications and has been an important subject of research for over twenty years in the communities of computer graphics, geometric modeling and computational geometry. In recent years, this research field has expanded significantly due to the improvements of 3D acquisition devices and algorithms (see Chapter 3). In this chapter we propose a classification of the different existing surface reconstruction methods, with a particular attention on techniques that are viable for datasets entangled with noise and containing outliers. We will also try to highlight their advantages and their limits regarding the flexibility we wish to attain.

Surface reconstruction techniques can be classified into two main approaches: the combinatorial approach and the models fitting approach. A large number of combinatorial approaches try to establish adjacency relations between points of a sample [Dey, 2004, Cazals & Giesen, 2006]. Most of these approaches were developed by computational geometers, who provided important theoretical contributions to the problem of surface reconstruction, especially with the notions of ε-sampling and restricted Delaunay triangulations to a surface. The reader can refer to Chapter 2 for a reminder of these concepts that will be reused later on.

The second approach is based on the idea to approximate the sampled surface using a chosen model, on the basis of global or local assumptions regarding the shape to be reconstructed. During the last decade, the graphics community showed a strong enthusiasm for local implicit approaches.

4.1 Combinatorial approach

The aim of combinatorial approaches is to establish adjacency relations (or connectivity relations) between neighboring points on the sampled surface. To achieve this goal without any prior knowledge on the organization of the data set, one can rely on the topological information that can be deduced from the proximity relations between points under some assumptions regarding the density of the sampling. Therefore, the combinatorial approaches exploit geometric structures allowing to measure the proximity of points in space. Among these structures, the Voronoi diagram and its dual Delaunay triangulation, are especially popular in computational geometry. They present some interesting properties to solve the reconstruction problem.

In the early eighties, [Edelsbrunner et al., 1983] introduced the α-shapes model in two dimensions, establishing back then the first rigorous mathematical definition of the shape of a point sample in the plane. In three dimensions, [Boissonnat, 1984] was the first to observe that the Delaunay triangulation of a set of points sampled on the surface can contain a subset of facets that are a good approximation of the sampled surface. Many algorithms were brought to us since then, with the same general principle, that is, to filter the connectivity relations established by the Delaunay triangulation to keep only the relevant ones. However, the conditions and criteria to obtain a good approximation of the surface were not defined until much later.

Inspired by a theorem of [Edelsbrunner & Shah, 1997] stipulating that if every Voronoi face of dimension d which intersects the surface does so in a topological $(d-1)$-ball, then the restricted Delaunay triangulation is homeomorphic to the surface, [Amenta & Bern, 1998] proved that if $\mathcal{P}$ is an ε-sampling of a smooth surface with $\varepsilon \leq 0.1$, then the restricted Delaunay triangulation of $\mathcal{P}$ to $\mathcal{S}$ is homeomorphic to $\mathcal{S}$. Naturally, in the context of surface reconstruction, the surface $\mathcal{S}$ is unknown. Therefore, the large majority of combinatorial algorithms try to extract a reasonable estimation of the restricted Delaunay triangulation from the Delaunay triangulation of the input point sample. In practice, the sampling conditions are not always verified, for example in the case of surfaces that are not locally differentiable. Some algorithms identify the facets of the Delaunay triangulation whose dual Voronoi edge intersect an approximation of the surface, in particular using estimated tangent planes, whereas other methods are based on the idea that the surface to reconstruct separates the internal medial axis of the external medial axis of a solid. Both informations can be obtained from the poles of the Voronoi diagram.

The connectivity relations established by the Delaunay triangulation are global by nature as they involve all the sample points. Other combinatorial algorithms use a local approach by considering the relation of closest neighbor in the Euclidean distance sense. These algorithms use some adapted spacial localization data structures to obtain efficiently the nearest neighbors of a given point.

In the following sections, we will present in more details the principal combinatorial methods. We have chosen to divide them into three main categories depending on their nature: the volumetric methods, the surface growing methods and the methods based on the Delaunay triangulation using poles.

4.1.1 Volumetric methods

The volumetric surface reconstruction method was introduced by [Boissonnat, 1984] through a **Sculpturing** technique. Starting from the convex hull of the point sample (the external facets of the Delaunay triangulation), cells are removed iteratively according to a shape and size priority criterion of the external facets, as well as topological constraints. The cell removal process is controlled by topological constraints, *i.e.* by prescribing the genus of the surface of the resulting solid. In particular, these rules limit the algorithm to reconstruct surfaces of zero-genus. Another major limitation is that the global heuristic governing the reconstruction process makes the result dependent on the order in which the cells of the Delaunay triangulation are removed. In some cases, the algorithm can stop prematurely, whereas another sequence of operations could have lead to a better result [Veltkamp, 1995]. Some algorithms that are based on sculpturing enabling topological modifications were proposed, but remain dependent on the order of execution of operations [Attene & Spagnuolo, 2000].

More recently several algorithms were based on the definition of ordering relations of the Delaunay triangulation cells rather than on a global heuristic based on local criteria. The process is then no longer subject to topological conditions, and the result depends no longer on the

traversal order of the Delaunay triangulation cells. The **Wrap** of [Edelsbrunner, 2003], the **Flow Complex** introduced by [Giesen & John, 2003] and the **Geometric Convection** algorithm of [Chaine, 2003] fall into this category. To put it briefly, these algorithms are interested in the gradient field ∇d of the distance function

$$d : \mathbb{R}^3 \to \mathbb{R},\ x \mapsto d(x) = \min_{p \in \mathcal{P}} \|x - p\| \ ,$$

induced by a point sample $\mathcal{P}$ of the surface. These algorithms are also interested in the relation this distance function has with the relative positioning of the Voronoi diagram of $\mathcal{P}$ and its dual Delaunay triangulation.

In the **Wrap** algorithm, Edelsbrunner defines a *flow relation* on the set of cells of the Delaunay triangulation of the sample points through the gradient field ∇d. The reconstructed surface produced by the Wrap algorithm is a subset of the Delaunay triangulation facets forming the boundary of the union of *stable manifolds* of a subset of maxima of d, consisting of the all points that converge towards these maxima following the flow induced by ∇d. The **Flow Complex** algorithm is a recursive structure of stable manifolds which provides a similar reconstruction to that of the Wrap algorithm. More recently, [Dey et al., 2005] proved that it is possible to generate a correct reconstruction from the Flow Complex under the hypothesis that $\mathcal{P}$ is an (ε, δ)-sampling with $\frac{\delta}{\varepsilon}$ fixed.

The sculpturing process can also be interpreted as the evolution of a surface through the Delaunay triangulation of the point sample. In the **Geometric Convection** algorithm of [Chaine, 2003] the sculpturing process is guided by a physical convection model introduced by [Zhao et al., 2001]. The algorithm contracts a surface encompassing the sample points under the influence of the gradient field $-\nabla d$ until it reaches a stable state. The stable state being based on the Delaunay triangulation and characterized by an *oriented Gabriel property*, *i.e.* for all facets of the reconstructed surface, the diametrical half-sphere oriented towards the inside of the surface is empty of sample points.

4.1.2 Surface growing approaches

The surface growing methods build a triangulated surface incrementally from the boundaries of an initial surface. The surface can either be directly constructed by considering a global topological criterion, or as a result of a post-processing step of a set of facets.

In his work [Boissonnat, 1984] proposed a second method that reconstructs incrementally a triangulated surface of a point sample by growing the mesh starting from some seed facets. A contour is propagated from the boundary of the current surface while considering local geometric properties to determine the new triangles to form. Given an edge of the contour, the neighboring points to this edge are projected on a locally estimated tangent plane. The next facet is built from this contour edge and a neighboring point. The later is chosen in a way that the angle at the vertex in the new facet opposite to the contour edge is maximized when this facet is projected on the tangent plane. The process is repeated until the surface is closed and that all points are considered. New seed facets can be created if the surface to be reconstructed has more than one connected component. This method does not use the Delaunay triangulation, however the author pointed out that the facets could be chosen from the Delaunay triangulation. This algorithm reconstructs a consistent surface under some restrictive topological conditions.

α-shapes The concept of α-shapes was first introduced by [Edelsbrunner & Mücke, 1994]. It is an approach to formalize surface reconstruction from 3D point sets. The *alpha*-shape is a generalization of the convex hull and a sub-complex of the Delaunay triangulation. Given

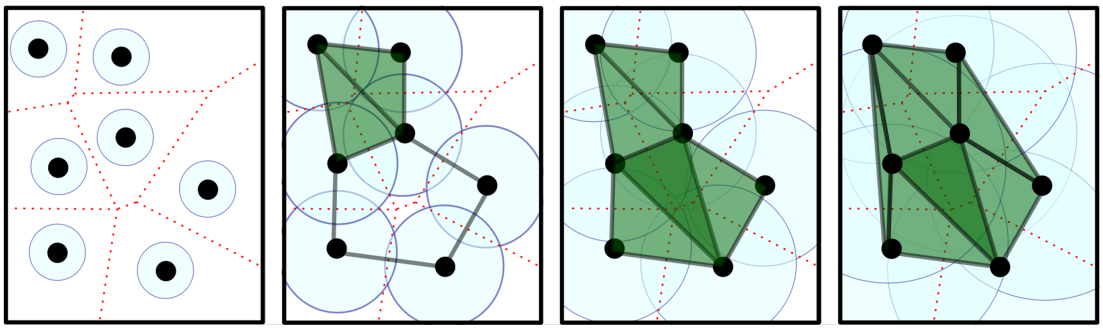


Figure 4.1: Reconstruction using α-shapes in 2D. The input points (black) with α-balls centered on them. From left to right: parameter α is increasing. The black edges form the α-complex and the α-shape is the boundary of the green area.

a finite point set $\mathcal{P}$, a family of "shapes" can be derived from the Delaunay triangulation of the point set; a real parameter α controls the desired level of detail through α-*balls* which are open balls with radius α (see Figure 4.1). The α-shape degenerates to the point-set $\mathcal{P}$ when $\alpha \to 0$ and to the convex-hull of $\mathcal{P}$ when $\alpha \to \infty$. The α-shape method consists of three steps: first, a Delaunay triangulation of $\mathcal{P}$ is computed, then the α parameter is selected and at last, the simplexes that are to be included in the reconstructed shape are identified. This approach allows fast, accurate, and efficient calculations of volume and surface area. One major drawback is that the input points need to be uniformly sampled. The α-shapes can be extended to deal with non uniform point sets [Edelsbrunner, 1992].

Several other algorithms based on the principle of incremental construction by advancing front were developed. The **Ball-pivoting** algorithm of [Bernardini et al., 1999], inspired by the α-shapes of [Edelsbrunner, 1992, Edelsbrunner & Mücke, 1994], builds the surface incrementally by "rolling" a ball on the sample points. The surface is initialized from an oriented seed facet such as a fixed radius ball passing through its three vertices is empty of sample points. The ball pivots around one of the edges of the facet until it reaches another point, forming a new facet. The process is repeated for all the edges of the current surface boundary. Once all edges have been visited, some new seed-facets can be added if not all points were reached. The algorithm produces a correct triangulation if the point sample is uniform. To handle non-uniform point samples, multiple passes are required with variable ball radii. Also, the algorithm requires an input point sample with oriented normals.

A more robust and flexible algorithm that filters the Delaunay triangulation of an unorganized point sample was designed by [Cohen-Steiner & Da, 2004]. The growth criterion is based on both distance and angle measures to choose the best facet at each step. These algorithms provide a guarantee that the reconstructed surface is a combinatorial 2-manifold, however they do not provide a guarantee on the approximation quality.

A drawback of maintaining a global topological criterion throughout the growth process is the risk to derail the reconstruction process in case of an error. An alternative approach is to apply the growth process to a subset of previously reconstructed facets or to selected facets according to some geometric criteria without them constituting necessarily a coherent surface. The surface growth process tries then to extract the surface to be reconstructed from this subset.

In their approach, [Ohtake et al., 2005b] propose to carry out the surface growing process by forming the dual facets of a cover of the point sample with spheres. First, they start building a spherical cover whose radius is linked to the local curvature, and where for each sphere a repre-

sentative point of the neighborhood of points it contains is computed. Adjacency relations are then established between representative points by examining the covering spheres intersection, then a surface mesh is obtained by filtering these relations in a way that the incident facets to a vertex form a plausible topological disk.

The approach of [Gopi et al., 2000] consists of approximating the surface around the neighborhood of each point of the sample by a two dimensional local triangulation. The algorithm proceeds in three stages, for each point $p \in \mathcal{P}$. First, an oriented tangent plane is estimated locally at p, the normals are then used to locally estimate the principal curvatures. The set of points located in a ball centered in p and whose radius depends on the local estimated curvature are then selected and projected on the tangent plane to p. A two dimensional Delaunay triangulation of the projected neighboring points to p gives a first set of candidate triangles incident to p, also called the *Delaunay neighbors* of p. Three points of the point sample form a facet of the reconstructed surface if they are mutually contained in their Delaunay neighborhoods.

A Delaunay triangulation filtering algorithm based on the extension of the Gabriel complex was introduced by [Adamy et al., 2002]. The *Gabriel complex* is a subset of the Delaunay triangulation that contains the facets whose smallest circumspheres are empty of sample points. The authors define an order on the Gabriel facets relative to the largest empty ball through their vertices. A facet has a higher priority and is more likely to belong to the reconstructed surface if it has one side of its supporting hyperplane where the points are far. The facet selection algorithm builds a topological disk in every sample point from the ordered set of Gabriel facets. This set of selected facets is then filtered to extract a surface. No guarantee is provided on the result in three dimensions.

4.1.3 Methods using poles

The algorithms described so far are mainly based on local criteria to determine or select valid connectivity relationships (except for the flow algorithm). Another family of algorithms exploits the fact that a solid object can be seen as an infinite union of maximal balls centered on its medial axis. For a sufficiently dense sample of points on a surface bounding a solid, a subset of Voronoi vertices are close to the medial axis of this solid object. This global information has been exploited both in volumetric algorithms and algorithms with a surface growing process.

More formally, the set of *poles* is a subset of Voronoi vertices. At most two poles can be extracted for each Voronoi cell $V(p)$, meaning that to each point $p \in \mathcal{P}$ corresponds at most two poles. Let $V(p)$ be a bounded Voronoi cell, the first pole v_1 is the Voronoi vertex in $V(p)$ with the largest distance to the sample point p whereas the second pole is the Voronoi vertex v_2 in $V(p)$ furthest away from p in the opposite half-space of v_1, *i.e.* such that the *vertex-pole* vectors $\overrightarrow{pv_1}$ and $\overrightarrow{pv_2}$ form an angle larger than $\frac{\pi}{2}$. See Figure 4.2 for an illustration in the two-dimensional and three-dimensional cases.

Crust The crust algorithm, or *Voronoi filtering*, was designed by [Amenta & Bern, 1998, Amenta, 1999]. The algorithm considers the enrichment of the Delaunay triangulation of the input point set $\mathcal{P}$ with all the *poles* Q. All facets of $D(\mathcal{P})$ intersected by the medial axis are eliminated if the poles are inserted in $D(\mathcal{P})$. The facets of the Delaunay triangulation $D(\mathcal{P} \cup Q)$ connecting three points of $\mathcal{P}$ are then likely to belong to the restricted Delaunay triangulation to the surface. This set of facets does not contain the final surface systematically, but it contains a surface that is homeomorphic to $\mathcal{S}$ if $\mathcal{P}$ is an ε-sampling with $\varepsilon \leq 0.1$. The final surface is obtained by applying a second filtering process on these candidate facets, starting by eliminating the triangles whose normals are far from the vertex-pole vectors. The remaining

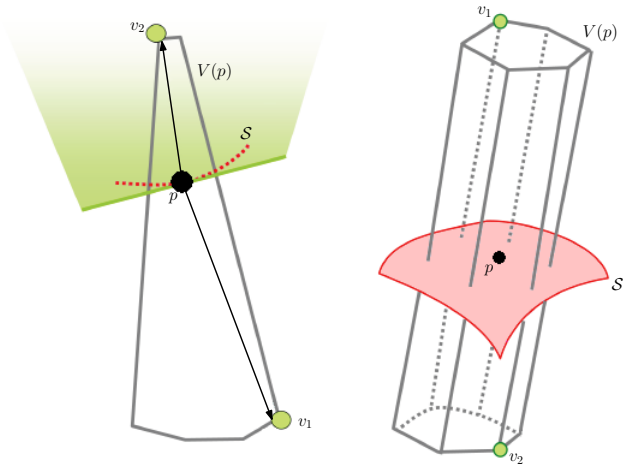


Figure 4.2: The poles. Voronoi cell $V(p)$ of sample point p ad two poles v_1 and v_2. Left: 2D case; right: 3D case.

facets are then oriented consistently, and a surface is extracted. Note that the crust algorithm was one of the first algorithms to provide theoretical guarantees.

Cocone The Cocone algorithm was designed by [Amenta et al., 2000] as an improvement over the Crust algorithm. Similarly, the algorithm computes a set of candidate facets and then extracts a surface. This time the candidate facets are selected directly from the Delaunay triangulation of the point sample using a criterion related to the geometry of the Voronoi cells. The *co-cone* at a point p is defined as the intersection of the Voronoi cell of p, with the complement of a double cone with apex p and fixed opening angle ($\approx \frac{\pi}{2}$) around the approximate normal at p. The later can be approximated by the vertex-pole vector. See Figure 4.3 for an illustration in the two-dimensional and three-dimensional cases. A facet is candidate if its dual Voronoi edge intersects the co-cones of its three vertices. If the sampling is dense enough, these candidate facets lie close to the original surface and the sampled surface can be extracted from the candidate facets. The algorithm's output is a surface homeomorphic to the original surface if the input sampling is an ε-sampling with $\varepsilon \leq 0.06$.

The original version of the Cocone algorithm is very sensitive to the sampling conditions, that are hard to meet with practical data. For example, the estimated normals may not be correct, leading to a wrong choice for the facets. Several improved versions of this algorithm were proposed. The Cocone algorithm was used to detect undersampled regions and borders in [Dey & Giesen, 2001], and was then extended to guarantee the construction of watertight surfaces [Dey & Goswami, 2003]. The same algorithm, associated to an oversampled regions detection method [Dey et al., 2001a] was used to elaborate a point sample simplification algorithm through undersampling [Dey et al., 2001b] A version of the Cocone algorithm was developed to handle large point data sets [Dey et al., 2001c]. More recently, a robust to noise version of the Cocone algorithm was studied [Dey & Goswami, 2006] with proves for Gaussian noise model.

Power Crust The Power Crust algorithm introduced by [Amenta et al., 2001] is an extension of the Crust algorithm that combines concepts of medial axis, Voronoi diagram and power diagram. The Power Crust algorithm is based on the observation that any point of a sampled compact surface is on the interface of two medial balls: the outer ball to the surface and the

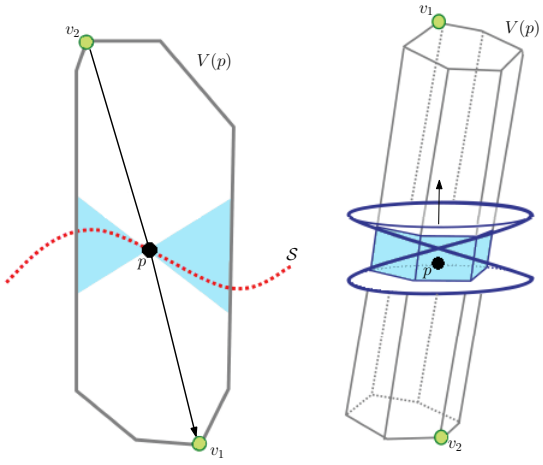


Figure 4.3: The co-cone [Amenta et al., 2000] of a sample point p on a curve in 2D (left) and a surface 3D (right) together with its Voronoi cell $V(p)$ and poles v_1 and v_2.

inner ball. The medial balls can be approximated by the circumscribing balls of the Delaunay cells dual to a pole, called *polar balls*. Amenta *et al.* have developed local criteria to distinguish *outer* from *inner* polar balls and they developed a global heuristic to differentiate inner balls from outer balls. If two adjacent polar balls intersect deeply, then the two corresponding poles are likely to be from the same side of the surface. Whereas if two balls intersect shallowly, one of the two cells is internal and the other external. The Power Crust algorithm computes the power diagram of the poles weighted by the radius of their polar balls. The reconstructed surface is the set of facets of the power diagram whose dual edges have an inner pole vertex and an outer pole vertex.

The classification idea based on analyzing the polar balls has been reused later on in other algorithms. In their work, [Dey & Goswami, 2006] showed that the criteria based on the intersection angle between polar balls suggested in the Power Crust algorithm, may fail in the presence of noise. They however show that polar balls in these regions have small radii, thus can be detected. Another algorithm derived from the Power Crust, called the **Eigen Crust** was proposed by [Kolluri et al., 2004] in order to produce watertight surfaces. The Eigen Crust algorithm handles undersampling, noise and outliers by using a spectral partitioning of the graph of poles, to obtain a robust tetrahedra classification. A simple modification of the Power Crust algorithm was suggested by [Mederos et al., 2005] improving the robustness of the original algorithm to point samples corrupted with Gaussian noise, with the guarantee to produce a surface homeomorphic to the sampled surface.

4.2 Models fitting

The surface reconstruction methods by models fitting seek to constrain an a priori determined global or local mathematical model of a surface to minimize the gap between the model and the data. Therefore, the reconstruction process comes down to an optimization problem. The surface can be constrained to pass through the sample points, we speak of *interpolation*, or close to the sample points, thus we speak of *approximation*.

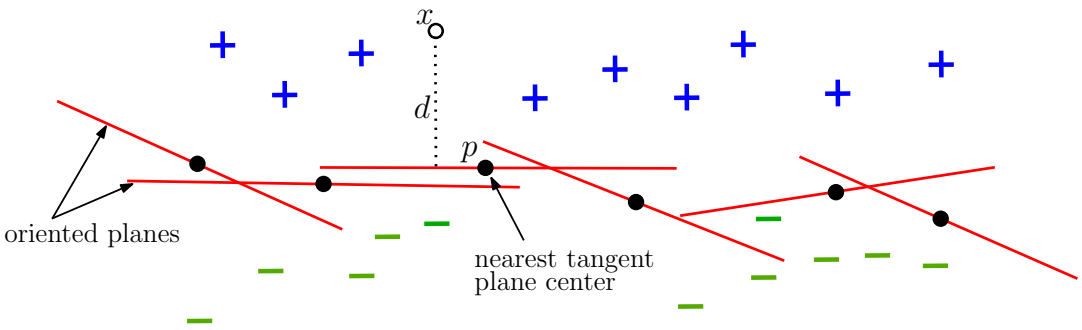


Figure 4.4: Signed distance function. For a given point x, its nearest neighbor p in the set of input points (black) is found, then the distance d to the estimated tangent plane at p is computed.

The model can be defined as a function depending on a finite set of parameters. The objective is to find the values of the models parameters to interpolate or approximate the input data, through a local or global approach; in this case we speak of *parameters fitting*. This was already discussed in the previous section namely when local differential properties of the sampled surface had to be estimated, as for example the normals, by tangent planes, or the curvatures, using quadrics. More generally, a model can be represented by a parametric or implicit function. In this section, we focus specifically on implicit surface models, which have fewer constraints and are more flexible to reconstruct complex shapes through a local approach, taking advantage of the ability to combine them easily. The ability of a model to provide a good approximation of the original surface is in the choice of appropriate basis functions, which involves making assumptions on the properties of the surface to be reconstructed, in particular of surface continuity. The nature of these functions and of the error to minimize, as well as the sampling conditions make the fitting process more or less complex and reliable. To output a triangulated mesh, these methods are often required to apply an isosurface extraction algorithm such as the classic Marching cubes algorithm [Lorensen & Cline, 1987], its feature preserving variant [Kobbelt et al., 2001] or the Delaunay refinement strategy (see Chapter 5 for more details on mesh generation algorithms). Alternative surface reconstruction approaches consider fitting *deformable models*, for which the reconstruction problem is solved iteratively by deforming successively an initial continuous or discrete surface.

4.2.1 Reconstruction by fitting an implicit surface model

The problems boils down to interpolate or approximate the data points with the aid of a continuous function $f : \mathbb{R}^3 \to \mathbb{R}$ whose zero level set defines a surface close to the sampled surface. To represent complex shapes, the function is usually constructed as a combination of simple basis functions, using for example low degree implicit functions [Ohtake et al., 2005a]. The surface to be reconstructed may be globally approximated by combining functions that locally approximate the surface, or by weighted sums of radial basis functions. Local methods typically employ combinatorial structures to decompose the data in space or to establish proximity relations between points. Many methods rely on the estimation of the surface differential properties, needing to have a neighborhood structure on the sample points. Most often, the neighborhood relations are chosen as the nearest neighbor relations in the Euclidean distance sense. Some local approximation methods build a distance function to an implicit surface out of distance functions to locally estimated tangent planes at each point, whereas other approximation methods consider the surface locally as the graph of a function defined on a parametric

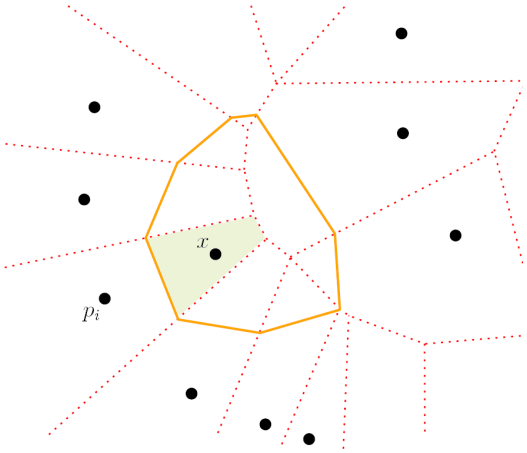


Figure 4.5: Sample point x has eight natural neighbors. The orange colored cell is created when x is added to the Voronoi diagram depicted in red. The normalized surface of the green area corresponds to the coordinate of x according to p_i.

plane domain.

Signed distance functions The first implicit surface reconstruction method presented to the graphics community was designed by [Hoppe et al., 1992]. This method estimates a signed distance function $d : \mathbb{R}^3 \to \mathbb{R}$ that is supposed to approximate the distance between a point $x \in \mathbb{R}^3$ and the original surface. The zero level set of this function $d(x) = 0$ defines the reconstructed surface implicitly. The distance of a point x to the surface is approximated by the distance between x and the tangent plane associated to the closest sample point in space. The sign of the distance function depends then on what side of the tangent plane x lies. The tangent plane at a sample point $p \in \mathcal{P}$ is estimated in the least squares sense from the principal components analysis (PCA) of the coordinates of this point p and its k nearest neighbors (see Figure 4.4). The normal vector direction is given by the direction of minimal variance of the position of points in this neighborhood. The tangent planes orientation is determined at a later time on a global scale so that the tangent plane associated to two close points have a similar orientation. The adopted strategy is to propagate normal orientation along a minimum spanning tree in the sense of the Euclidean distance (see Appendix A).

The reconstructed surfaces have $\mathcal{C}^0$ continuity. This estimation and consistent orientation technique of the tangent planes was later used on by various reconstruction methods that require oriented normals at each sample point.

Such algorithms require a uniform sampling at least locally since otherwise the k-neighborhood can be spatially biased in the case of oversampling for example, resulting in a poor estimation of the tangent planes.

Natural neighbors interpolation of signed distance functions In their work [Boissonnat & Cazals, 2000] proposed a reconstruction method through the sign of a function $f : \mathbb{R}^3 \to \mathbb{R}$ whose *zero level set* defines an implicit surface approximating the sampled surface. The function f is built by interpolating the natural neighbors of *signed distance functions* defined around each sample point. The *natural neighbors* of a point x are defined as the neighbors to x in the Delaunay triangulation of $\mathcal{P} \cup \{x\}$. This information is of combinatorial nature and can be made quantitative using the so-called *natural coordinates*. The natural coordinate of a point x according to a point $p_i \in \mathcal{P}$ is the normalized measure of the region

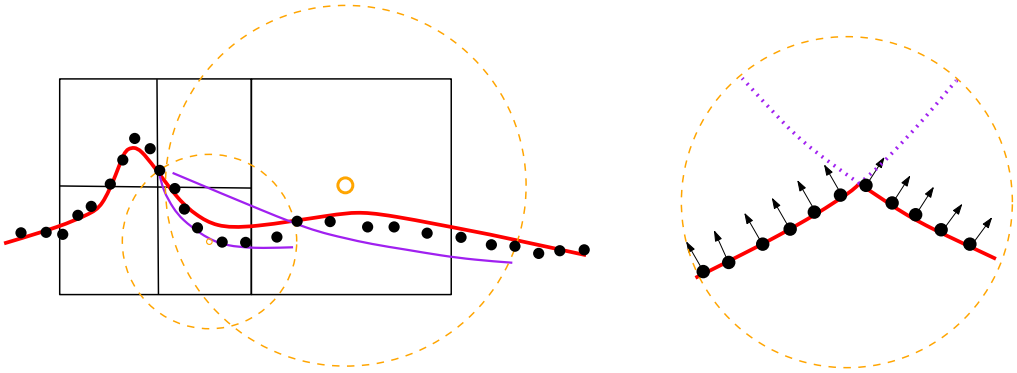


Figure 4.6: Multi-level partition of unity implicit [Ohtake et al., 2003a]. Left: an octree is computed according to the distribution of the input data points (black). For each cell of the octree, a quadric surface (purple) is fitted to the points contained in a ball centered at the cell center. The type of quadrics fitted is chosen accordingly to the distribution of points. Then, the local reconstructions are "blended" to obtain the surface (red). Right: illustration of the case where an edge is detected by normal clustering.

withdrawn from p_i if x is added in the Voronoi diagram (see Figure 4.5). The function f at a point x is then defined as the mean signed distance from x to the tangent planes of the natural neighbors of x, thus requires a coherently oriented normal at x. The tangent planes can be estimated through the poles, plus a heuristic for determining a consistent orientation [Boissonnat & Cazals, 2001].

This method works in any dimension and is suitable for surfaces of arbitrary topology and non-uniform sampling.

More recently [Mullen et al., 2010] designed a modular framework for robust surface reconstruction from unorganized, noisy and outliers-ridden point sets. Their idea is to use an unsigned distance function to the input data followed by a global stochastic signing of the function. The isosurface reconstruction is then obtained via a sparse linear solve.

Partition of unity The partition of unity approach decomposes the data domain into smaller overlapping sub-domains, where the reconstruction problem is locally resolved. Initially, the data are approximated in each sub-domain separately by functions $f_i : \mathbb{R}^3 \to \mathbb{R}$, then the local solution are "blended" together using a locally supported weighted sum:

$$f(x) = \sum_{i=1}^{n} w_i(x) f_i(x)\,,$$

where n is the number of sub-domains. The functions $w_i : \mathbb{R}^3 \to \mathbb{R}$ are weighted functions having compact support, centered on the sub-domains, and such that they sum to one uniformly on the whole domain.

One of the earliest methods of surface reconstruction using this formulation is the natural neighbors interpolation of signed distance functions method by [Boissonnat & Cazals, 2000] described in Section 4.2.1. The tangent planes defined at each sample point correspond to the local reconstructions, and the Delaunay triangulation of the point sample is the partition of the data domain. Methods implementing approximations of higher degree have been developed subsequently. [Ohtake et al., 2003a] proposed to reconstruct the surface locally using quadric functions or piecewise quadric capturing the local shape of the surface. The partition of unity

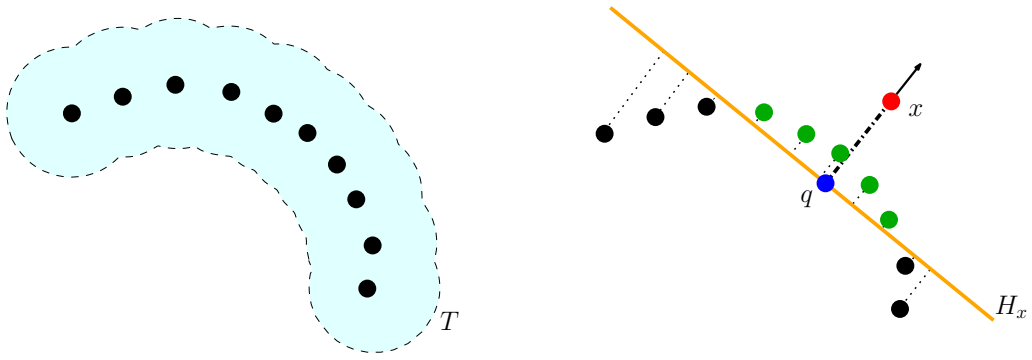


Figure 4.7: Moving least squares. Left: the tubular neighborhood T of the input points (black). Right: A local reference plane H_x (orange) for the point x (red) is determined by minimizing a local weighted sum of squared distances of the points (green) to the plane H_x. Point q (blue) is the projection of x onto H_x.

is built using an adaptive octree, a local study of the normals deviation controls the choice of surface model in each leaf-cell. This method is very effective on large sets of points and can control the level of detail of the reconstruction. However, it is highly sensitive to the quality of input data, especially to those of normals. Moreover, in order to detect sharp features, a clustering of normals is performed (see Figure 4.6). [Tobor et al., 2004] used the partition of unity approach with radial basis functions (RBF) to compute the local approximations, with a hierarchical decomposition on a grid. This method reduces the complexity of the problem of global reconstruction using RBF to handle large sets of points.

Recent attempts by [Nagai et al., 2009] at fitting smooth implicit functions through partition of unity show promising results when it comes to robustness to noise.

Moving Least Squares The Moving Least Squares (MLS) surface model introduced by [Lancaster & Salkauskas, 1981] is another local implicit surface reconstruction approach. The particularity of this model is that the function representing the global surface is not explicitly defined. Local approximations are only computed in the neighborhood of points where the function is evaluated [Shen et al., 2004]. In recent years, this technique has found some success in the graphics community, especially after the work of [Alexa et al., 2003], to model surfaces from point samples without maintaining explicit connectivity, *i.e. point set surfaces*. We refer the reader to the survey by [Kobbelt & Botsch, 2004] for more details on point-based techniques in computer graphics.

Given a set of input points and their oriented normals, [Levin, 1998, Levin, 2003] proposed an implicit surface model that approximates globally the data based on the MLS local approximation method. Levin defines an *MLS surface* (S_{MLS}) through a projection operator $\psi : T \to \mathbb{R}^3$ that projects all points located in a *tubular neighborhood* T, illustrated in Figure 4.7 left, of that surface onto itself. The MLS surface is therefore defined as the set of points of T that project on themselves:

$$S_{\text{MLS}} = \{x \in T | \psi(x) = x\}\,.$$

In this approach, the function that models the surface is not explicitly determined, it is defined implicitly through the projection operator. The heart of the method is a three steps projection procedure that computes the projection of a point $x \mapsto \psi(x)$ on a local polynomial approximation of the surface H_x in a neighborhood that depends on the position of $\psi(x)$. The first step determines a local *reference plane*. This reference plane is later used as a parametric domain

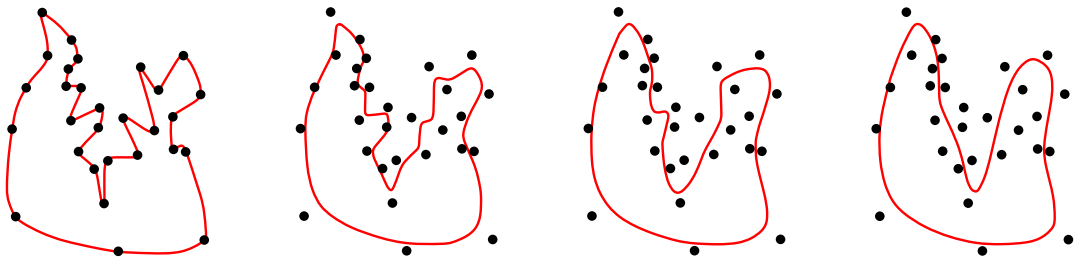


Figure 4.8: Radial basis functions. From left to right the value of the regularization parameter λ increases.

to compute the local polynomial approximation of the surface. The first step is the toughest as it requires to solve an iterative non-linear minimization problem to determine the direction of the normal at $\psi(x)$. The second step computes the local polynomial approximation, leading to solve a standard linear least squares problem where the polynome coefficients are determined by solving the system of linear equations. The last step computes the position of the projected point on the local surface. Figure 4.7 illustrates the moving least squares approach in 2D.

Levin has shown that under some sampling density and continuity conditions, determined by the weight functions used in the first step, the resulting surface is a C^∞ manifold. The method is naturally robust to Gaussian noise, but the projection process can fail if the local geometry is not sufficiently sampled or if some outliers lie in the point sample.

Several authors revisited the original model of Levin to provide surface reconstruction algorithms that are robust to noise [Xie et al., 2003], may reconstruct a multiresolution surface approximation [Mederos et al., 2003], or that cope with outliers and represent piecewise-smooth surfaces [Fleishman et al., 2005] (please refer to Section 7.1.2 for a detailed related work on piecewise smooth surface reconstruction). [Scheidegger et al., 2005] designed a specific algorithm to triangulate an MLS surface through the Marching Cubes method. Also, [Boissonnat & Oudot, 2004] applied their surface sampling method to MLS surfaces to produce optimal triangulated surfaces based on the model of [Adamson & Alexa, 2003].

Several authors have looked at the overall validity of the MLS surface model. In their work [Scheidegger et al., 2005] analyzed the stability of the original MLS projection operator depending on the proximity of points to the MLS surface. They also proposed an explicit definition of the MLS surfaces in terms of critical points of an energy function defined in a vector field. [Kolluri, 2008] proved the validity of the model designed by [Adamson & Alexa, 2003] for sufficiently dense and uniform point clouds. Another extension of the MLS model suggested by [Dey & Sun, 2005] provides reconstruction guarantees for non-uniform sampling conditions.

Radial basis functions Thirty years ago, in an extensive survey, [Franke & Nielson, 1980] identified Radial Basis Functions as a relevant method to solve interpolation problems from scattered data. Given a set of n points $\{p_i\} \subset \mathbb{R}^3$ to which are attached scalar values $\{h_i\}$, the aim is to approximate a function $f : \mathbb{R}^3 \to \mathbb{R}$ such that $f(p_i) \approx h_i$ for all pairs (p_i, h_i), while minimizing an a priori chosen error E, that can be chosen for example as follows:

$$E(f) = \sum_{i=1}^{n} (f(p_i) - h_i)^2 + \lambda R(f)\,,$$

where the first term corresponds to a least squares error, R is the regularity prior term and λ is a positive scalar value, called *regularization parameter*. The later allows to tune the degree of smoothness. When $\lambda \to 0$, the surface closely fits the points. Contrariwise for large values

of λ, the surface approximates the data points by increasing the regularity of the function. Figure 4.8 illustrates an example in 2D.

As part of the RBF approach, the function that minimizes the functional E is defined from a class of basis function $\phi : \mathbb{R} \to \mathbb{R}$ with local or global support depending on the choice of E, as a linear combination:

$$f(x) = \sum_{j=1}^{m} w_j \phi(\|x - c_j\|) \,,$$

where $\{c_j\}_{j=1\ldots m}$ denotes a set of m constraint points, or *centers*, and $\{w_j\}$ denotes a set of weights attributed to each basis function. The surface reconstruction problem is reduced to determine the vector $w = \{w_1, \ldots, w_m\}$ by solving a system of linear equations.

The zero level set of the function f defines an implicit surface that interpolates or approximates the constraint points. For a random point sample of a surface, all points are associated to constraints such that $f(p_i) = 0$. To avoid the trivial solution $w = 0$, additional constraint points with non-zero values must be chosen from outside and/or inside the surface. These constraints may, for example, be obtained by shifting along the data points oriented normals.

Early work in the graphics community on this surface reconstruction approach where presented by [Savchenko et al., 1995] followed by [Turk & O'Brien, 1999, Turk & O'brien, 2002]. The methods that are based on RBF with global support are particularly robust, however they require solving dense systems of linear equations which makes it difficult to process samples with more than tens of thousands of points ($O(n^3)$ complexity). To accelerate the evaluation and handle large data sets, different approaches have been studied. In their work [Carr et al., 2001] reduce the number of centers by an iterative selection procedure and use a quick evaluation method of RBF, *i.e. Fast Multipole Method*, whose principle is to evaluate the basis function approximatively beyond a given distance to the centers. [Ohtake et al., 2003b] first and then [Morse et al., 2005] rely on compactly supported RBFs with adaptive support to handle noise. However, the choice of adapted support radii is difficult for non-uniform point samples. Also, this type of solution poses a scalability problem to handle large data sets. More recently [Samozino et al., 2006] designed a novel surface reconstruction algorithm using compactly supported RBFs that bypasses these scalability limits by placing the RBF centers on the medial axis estimated from the Voronoi diagram of the samples.

Another way to deal with the performance issues of the RBF approach is to proceed by partition of unity. Both [Tobor et al., 2004] and [Ohtake et al., 2004] decompose the data into a hierarchy of subdomains where the interpolation problem is solved locally. The local reconstruction are blended at each node of the hierarchy, the root contains the function that corresponds to the global reconstruction.

Indicator functions Given a set of input points and their oriented normals, an alternative to the aforementioned functions is the indicator function. The later is valued one inside the object and zero outside. [Kazhdan, 2005] noted that the gradient of the indicator function χ is zero everywhere, except on the surface where it is equal to the surface normal (see Figure 4.9). The sparse input normal field $\vec{n}$ estimated using an octree structure is extended everywhere and the alignment of the indicator function with this normal field is computed in the least squares sense by minimizing the functional:

$$\int_{\Omega} \|\vec{n}(x) - \nabla\chi(x)\|^2 \, dx \,,$$

where the indicator function χ is relaxed to take any real value. Applying the divergence

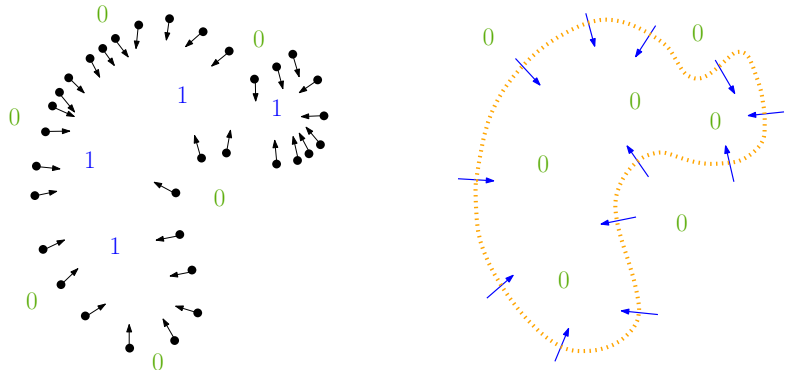


Figure 4.9: Poisson surface reconstruction [Kazhdan et al., 2006]. Left: the set of input points and their oriented normals (black). The value of the indicator function χ is zero outside and one inside the object. Right: the indicator gradient $\nabla\chi(x)$ is non-zero on a neighborhood of the input points.

operator leads to a Poisson problem:

$$\Delta\chi = \operatorname{div}\overrightarrow{n}\,.$$

Initially solved in the Fourrier domain by a *Fast Fourrier Transform* (FFT) in [Kazhdan, 2005], locally supported RBFs are used in [Kazhdan et al., 2006] on the cells of an adaptive octree for efficiency and produces great results making the method until now very competitive as the global $\mathcal{L}^2$ minimization naturally brings resilience to noise.

A variant of the Poisson reconstruction called *adaptive Fourrier based* surface reconstruction was designed by [Schall et al., 2006, Schall et al., 2007]. This approach is based on the partition of unity and performs an error guided subdivision of the input data points. The space decomposition is based on an octree. A local solution is computed as a local characteristic function for the points inside the cell using Kazhdan global FFT approach. Since the points inside a cell do not represent a solid, the characteristic function may not be computed using the Poisson global FFT approach. To avoid creating surface parts which are not represented by points, the solution is to subdivide the octree cells if the resulting local approximation inside the cell is not accurate enough. By iterating this procedure, overlapping local characteristic functions are computed for each octree leaf of each part of the input with a user defined accuracy parameter. The final reconstruction is then obtained by combining the local approximations using the partition of unity approach. The algorithm runtimes are better than the ones of Kazdhan, however the characteristic function is only determined close to the surface and not for the whole volume.

This family of implicit approaches can be limited by their sensitivity to noise, outliers or non-uniform sampling or even simply the lack of reliable and consistent normal information.

Graph cut based reconstruction Building signed distance functions requires that the set of input points comes with consistently oriented normals. If the later are not provided, the estimation of consistent orientations puts up many problems. Methods based on signed distance functions often pose problems of topological noise, *i.e.* the reconstructed surface has not the same genus as the original surface. This also poses problems for the meshing algorithms.

Recent research on combinatorial energy minimization has shown that globally optimal solutions to discrete volumetric segmentation problems can be found efficiently by reformulating them into a maximum flow/minimum cut problem of a specific spacial graph structure.

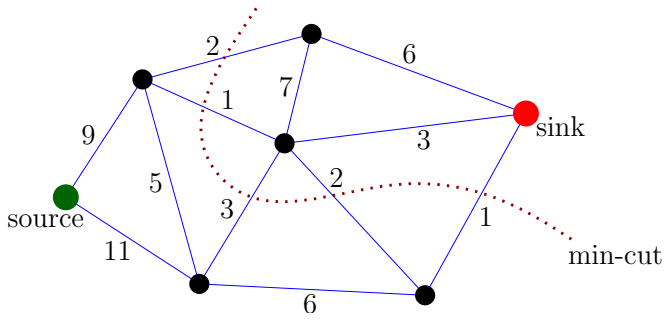


Figure 4.10: Two dimensional graph min cut. Given a graph with weighted edges, find min-cut between source and sink nodes.

An undirected graph $\mathcal{G} = \langle V, E \rangle$ is defined as a set of nodes V and a set of undirected edges E that connect these nodes. An example of such a graph is shown in Figure 4.10. Each edge $e \in E$ of the graph is assigned a non-negative weight w_e (*cost*). There are also two special nodes called the *sink* and the *source*. A cut is a subset of edges $C \subset E$ such that the sink and the source become separated on the induced graph $G(C) = \langle V, E \backslash C \rangle$. Each cut has a cost which is defined as the sum of the costs of the edges that it severs:

$$cost = \sum_{e \in C} w_e \, .$$

The *minimum cut* problem on a graph is to find a cut that has the minimum cost among all cuts.

In their work [Hornung & Kobbelt, 2006] proposed a volumetric method for surface reconstruction from unstructured point sets that is based on an unsigned distance function to the set of points, defined over a regular 3D grid. The authors identify the voxels intersected by the surface using a graph structure embedded within the voxel grid and by implementing a combinatorial energy minimization method to localize the surface.

The algorithm starts by a volumetric diffusion phase in the grid that enables to construct an unsigned distance field in a narrow band around the points sample. The diffusion is effected by scanning the grid with a mean filter, having previously initialized to one the voxels containing points and the others to zero. A labeled graph is then built on the voxels located in this neighborhood. Each node of this graph is associated to a facet of a voxel and the edges connect the neighboring facets in the 8-connexity sense. Each edge receives a weight that is linked to the potential value of the traversed voxel. An algorithm, based on the minimal surfaces framework with graph cuts of [Boykov & Kolmogorov, 2003], computes a minimum cut that determines the optimal set of voxels traversed by a smooth approximation of the original surface.

Recently [Lempitsky & Boykov, 2007] propose a global optimization framework for 3D shape reconstruction from sparse noisy 3D data sets also based on the minimal surfaces framework with graph cuts of [Boykov & Kolmogorov, 2003]. In contrast to earlier local or banded optimization methods for shape fitting, they compute global optimum in the whole volume removing dependence on initial guess and sensitivity to numerous local minima.

4.2.2 Deformable models

The methods of this class deform an initial surface to obtain a good approximation of the set of input points. Intuitively, the surface is obtained from an elastic membrane on which is applied a deformation process to minimize an energy functional. Since the seminal paper of [Kass et al., 1988] on active contours many applications of deformable models that evolve under the sum of internal and external energies have been found in computer vision and graphics. Generally, the methods based on deformable models are interesting for their robustness, but require a good initialization and are prone to drop into a local minima. Hereafter we present two classes of methods that use an implicit surface model.

Level sets framework In the level set framework of [Dervieux & Thomasset, 1980] and of [Osher & Sethian, 1988], the time-varying model is represented as the zero level set of a function in $\mathbb{R}^3$ whose level sets are all evolved. More recently [Zhao et al., 2001] proposed a surface reconstruction method from an unorganized set of points whose principle is to contract a closed surface enclosing all input points. The surface is represented implicitly by the dynamic formulation of level sets.

More formally, a surface denoted $\tilde{\mathcal{S}}$ enclosing the whole set of input points is deformed to minimize an energy functional defined as a weighted area, where each surface element is weighted by its distance to the closest point sample.

$$E(\tilde{\mathcal{S}}) = \left[\int_{x \in \tilde{\mathcal{S}}} d^p(x) ds \right]^{\frac{1}{p}} , \; p > 0 \, ,$$

where $d(x)$ is the distance between $x \in R^3$ and its closest point in $\mathcal{P}$, and ds a surface element. The energy functional E is minimized to a local minimum by gradient descent by contracting the initial covering surface. At each step, every point x of the surface $\tilde{\mathcal{S}}$ evolves toward the interior of the surface, along the normal direction to $\tilde{\mathcal{S}}$ at point x denoted $\overrightarrow{n}(x)$ with a displacement speed that is proportional to $-\nabla d(x).\overrightarrow{n}(x) + d(x)\kappa$, where κ denotes the mean curvature. The first term $-\nabla d(x).\overrightarrow{n}(x)$ reflects the attraction of the surface $\tilde{\mathcal{S}}$ towards the data set, whereas the second non linear term $d(x)\kappa$ represents the tension of the surface. The surface evolution process runs until reaching an equilibrium state characterized by the following:

$$\nabla d(x).\overrightarrow{n}(x) + d(x)\kappa = 0 \, .$$

Zhao *et al.* compute an initial approximation of $\mathcal{S}$ taking only into account the gradient field $-\nabla d$ and use it as the initial surface $\tilde{\mathcal{S}}$; this initialization scheme amount to perform the convection of a covering surface in the gradient field $-\nabla d$. Provided that the initialization is close to the optimum, this method is fast and robust to noise. Through its evolution; the topology of the surface can also easily change, which is an important benefit of the implicit representation.

Evolution of an explicit deformable mesh guided by a scalar field [Sharf et al., 2006] presented recently a method based on an explicit deformable surface model, represented by a mesh, evolving in a scalar field. An initial triangulated surface included in the object volume is deformed locally by moving its vertices in the direction of their normals. The evolution of this surface is guided by signed distance field sampled on an adaptive grid defining an implicit representation of the targeted surface. Throughout this process, the displacement of the mesh vertices is constrained to maintain the coherence of the surface that evolves (Laplacian system). The authors control the reconstruction level of detail by considering several independent evolution fronts with different properties to adapt to local forms. The mesh is dynamically optimized according to the local geometry and the process terminates with an MLS projection phase to capture the local details.

4.3 A motive for further researches

Surface reconstruction from raw point clouds has received increasing attention in the past decade due to the ever broadening range of point cloud acquisition sensors and computer vision algorithms that provide little to no reliable attributes (see Chapter 3). The inevitable presence of noise and outliers makes the scientific challenge even greater, where any advance in this direction can directly benefit surface reconstruction from point sets with some attributes. A common line of attack to this problem involves filtering out outliers before resorting to a reconstruction method as mentioned in the preceding sections.

Usually, the set of input points is first oriented, *i.e.* normals are computed, or, signed, *i.e.* an inside/outside function is constructed based on the point set. However, outliers removal is hardly automatic, thus requires an interactive adjustment of parameters (see Appendix A.4 for an outlier removal algorithm). Moreover, determining reliable normals with consistent orientation from raw point sets can be as hard as reconstructing the whole surface itself: although there are several choices to reliably estimate normal directions [Mitra et al., 2004, Alliez et al., 2007], robust normal orientation is significantly more challenging, as recently recalled by [Huang et al., 2009].

It should be noted that most of the aforementioned methods target reconstruction from high quality data. Variational methods such as [Walder et al., 2005, Alliez et al., 2007], based on generalized eigenvalue problems are better able to extract smooth surface from noisy data; unfortunately robustness to outliers is forlorn and their current complexity does not scale well with dataset size. Consequently there is a need to explore alternative surface reconstruction solutions that would provide scalability as well as robustness to noise, outliers and possibly undersampling. This is typically the case of datasets generated from multi-view passive stereo methods. Indeed, in this thesis we will follow this trend and investigate the topic of surface reconstruction from 3D data generated from multi-view passive stereo methods in Chapter 6.

Recently, in the field of photogrammetry [Goesele et al., 2007, Furukawa & Ponce, 2007] and [Bradley et al., 2008] presented specifically designed heuristics to filter, densify and improve the quality of point clouds or depth maps from multi-view passive stereo and to make some of the most robust surface reconstruction techniques [Kazhdan et al., 2006] still applicable to these refined point clouds. This issue was also remedied for data coming from multi-view passive stereo by [Labatut et al., 2007, Vu et al., 2009] where reliable lines of sight are available thus making it easier to classify outliers and to generate hight quality 3D meshes. More generally, the more attributes we have the more robust the reconstruction can be.

In most sciences one generation
tears down what another has built,
and what one has established,
another undoes.
In Mathematics alone
each generation adds a new
storey to the old structure.

Hermann Hankel (1839-1873)

5

Automatic simplicial mesh generation methods

A large body of literature discussing the hard task of automatic simplicial mesh generation has emerged over the past thirty years. In the following we provide a quick overview of different methods for automatic simplicial mesh generation. These methods can be classified into three main categories: grid based methods, advancing front methods and methods that are based on Delaunay triangulations. We would like to emphasize that this classification is not exhaustive and/or strict, there also exist various other mesh generation techniques that have been developed although none of them seems to be intensively used today. It is nevertheless interesting to point out that Delaunay triangulations became popular in the numerical and graphical calculus community, where several hybrid methods emerged.

The hybrid methods usually combine frontal or hierarchical grid based methods to Delaunay triangulations to connect the vertices and generate a simplicial mesh. Namely, some of these hybrid methods use an advancing front strategy to generate vertices that are later connected using a Delaunay triangulation [Merriam, 1991, Marcum & Weatherill, 1995, Mavriplis, 1995, Frey et al., 1996, Frey et al., 1998]. Whereas other methods [Schroeder & Shephard, 1990] combine both hierarchical grid and Delaunay approaches to fully automatic mesh generation. The resulting algorithm provides the linear growth rate and divide-and-conquer paradigm of the octree method, with the simplicity and optimal properties of the Delaunay triangulation.

In this chapter, we will concentrate on the three main enumerated mesh generation categories. For a complete review of this topic we refer the reader to the following PhD thesis [Shewchuk et al., 1997], surveys [Owen, 1998, Bern & Plassmann, 2000, Du & Wang, 2006], and book [Frey & George, 2008].

5.1 What is a simplicial mesh?

A *mesh* is a cellular *complex* partitioning an input domain into a finite number of elementary cells. The definition of a complex implies that the cells are convex, and that any two cells are disjoint or share a common lower dimensional face. Moreover, the cells of a mesh are required to be elementary, meaning that they admit a bounded size description. Among meshes, *simplicial meshes* are the ones where all the elements are simplices. In this thesis we are mainly concerned by simplicial surface meshes in the three dimensional space, composed of simplices of dimension less than or equal to two.

Meshes are one type of representation that is coveted in many areas of science and engineering for their processing efficiency. We refer the reader to the following book [Botsch et al., 2010]

that discusses the whole geometry processing pipeline based on simplicial meshes. Also, we encourage the reader to explore the capacities of the broadly-useful mesh processing system by [Cignoni et al., 2008] that we used for multiple tests and rendering.

5.2 Spatial decomposition mesh generation

Mesh generation methods based on grids, quadtrees in 2D or octrees in 3D, were widely studied throughout the eighties. Many algorithms were brought to us since then, with the same general principle, that is, in the three dimensional case, cubes containing the object or the domain to be meshed are recursively subdivided until a defined size is met. The resulting cubes are then subdivided into simplices. However, around the boundaries of the object, these methods require extensive intersection computations and might generate irregular simplices. Although the resulting meshes were satisfying in most cases [Yerry & Shephard, 1983, Yerry & Shephard, 1984, Baehmann et al., 1987], no theoretical guarantees were provided on the quality of the produced meshes.

The first algorithm to provide theoretical guarantees is a polygon meshing algorithm proposed by [Baker et al., 1988]. In practice, they generate a triangular mesh using a basic uniform grid such that all mesh elements have angles bounded between 13° and 90°, with the exception of regions where the input polygon has intrinsically angles smaller than 13°. As designed, this algorithm does not provide any control on the number of mesh elements, however the authors propose to use more adaptive data structures to improve their algorithm, namely, by using quadtrees instead of uniform grids.

Another algorithm was proposed by [Bern et al., 1990] in order to mesh a polygon with guarantees on both the shape and size of the mesh elements. This idea was later reused and extended by [Mitchell & Vavasis, 1992] to produce meshes with guarantees in 3D using octree data structures, and more recently to any dimension by [Mitchell & Vavasis, 2000]. The algorithm of Mitchell and Vavasis fails however in practice to produce triangles with the guaranteed angles in the three dimensional case. An alternative approach, also based on octrees, was suggested by [Shephard & Georges, 1991] providing better results in practice without the theoretical guarantees on the size and shape of mesh elements.

The *Marching Cubes* mesh generation algorithm, proposed by [Lorensen & Cline, 1987], laid the foundation for the extraction of a polygonal approximation of an implicit surface. To put it briefly, the algorithm divides the three dimensional space containing the object into a hierarchical grid of cubes. Each cube is defined by its surrounding eight grid vertices, and the implicit function is evaluated for each of these eight vertices. If the sign of all eight values is equal, the cube is entirely inside or outside the object. In this case, the cube is not intersecting the surface of the object and therefore is not further processed. Cubes experiencing a sign change across their eight vertices do intersect the object surface. Since there are two possible signs for each vertex, namely, a negative sign for an interior point and a positive sign for an exterior point, there is a total of 256 possible cases for each cube. The Marching Cubes algorithm specifies how to generate polygons for each of these cases.

More recently, the *Extended Marching Cubes* algorithm designed by [Kobbelt et al., 2001] attempts to solve two major problems of the classic Marching Cubes algorithm that act directly on the quality of the generated meshes. On one hand, the computation of exact intersection points, for which they suggest using linear interpolation. On the other hand, approximating sharp edges and corners, for which they examine not only the exact intersection points, but also the normal vector at each point, and they use the combination of both to estimate the

location of sharp edges and corners. Another feature-preserving approach was proposed in [Ju et al., 2002]. It is able to generate accurate meshes with sharp features using an octree data structure, resulting in meshes with fewer faces.

5.3 Advancing front mesh generation

Another very popular family of simplicial mesh generation algorithms is the advancing front, or moving front method. The advancing front approach starts by building a mesh of the object boundaries using a sizing field. This *boundary mesh* is used as the initial front. The object is then incrementally filled with simplices that are made up of a front element connected to an existing vertex of the mesh or to an inserted vertex in the inner area bounded by the front. At each step of the algorithm, the front evolves dynamically and separates the meshed from the unmeshed regions of the object. Each time a simplex is added, the front is locally updated and the algorithm continues until the whole inside of the object is filled with simplices.

This approach was first introduced in two dimensions by [George, 1971] and studied extensively by [Lo, 1985, Löhner & Parikh, 1988, Lo, 1991, George & Seveno, 1994, Löhner, 1996], in two and three dimensions, among others. In practice, advancing front mesh generation methods require a suitable choice of starting boundary mesh to ensure that the boundary mesh allows: on one hand to generate a mesh of the object that respects the size criteria, on the other, to avoid that two fronts with elements of different sizes will meet at the center of the object, because then the method can't help but create mesh elements of poor quality.

Advancing front methods allow significant control over the quality of the mesh elements: once an element of the front is selected, the vertex used to create a new simplex is precisely chosen to form the simplex of best quality. If this optimal vertex is too close to an existing vertex, the simplex will be created using the most adapted vertex among the closest vertices. This produces simplices of very good quality with the exception of regions where the fronts meet. Unlike the aforementioned hierarchical grid mesh generation methods, the advancing front methods create very good elements in the vicinity of the object boundaries.

Advancing front methods were adapted, in two dimensions, to generate anisotropic meshes where the mesh elements quality does not rely anymore on a sizing field, but rather on metric tensors field [Lee, 1999]. However, in three dimensions, the advancing front methods might not always generate a simplicial mesh, especially when the geometry of the object is complex. This *front closeness* problem [Du & Wang, 2006] is stil an unresolved issue, although various heuristic approaches, such as element deletion and trial-and-deletion have been proposed [Rassineux, 1998].

The advancing front method was also declined to mesh surfaces [Lau & Lo, 1996, Lo & Lau, 1998]. At each step, the front grows in a tangent plane to the surface, and the new vertices are projected onto the surface. This method requires a lot of triangle-triangle intersection computations, to verify that the elements of the generated surface mesh do not intersect.

Recently, [Scheidegger et al., 2005] adapted in a novel way the advancing front algorithm to directly mesh point set surfaces (see Section 4.2.1 for a presentation of point set surfaces). Their algorithm produces high-quality triangular meshes with bounded error, even in the case noisy input data, and is capable of generating meshes with user-defined approximation errors.

More recently, [Schreiner et al., 2006] extended the approach in [Scheidegger et al., 2005] and proposed an advancing-front algorithm that produces high-quality triangular meshes from many

different types of source objects, including point-set surfaces. Their approach is based on defining a guidance field, based on local curvature information to determine the size of the triangles. They start by computing in advance a specific guidance field that allows the mesh generation scheme to adapt to areas of high curvature, shrinking the authorized triangle size as the advancing fronts grows towards such regions. The algorithm uses MLS (see Section 4.2.1 for a presentation of MLS) for projection and curvature computations on point clouds. Consequently, their method adaptively reduces the triangle sizes near sharp features and thus smooths sharp edges.

5.4 Delaunay refinement mesh generation

The third category of simplicial mesh generation algorithms is composed of all algorithms that refine a Delaunay triangulation. In such methods, an initial mesh is generated by simply computing the Delaunay triangulation of a set of points. This "coarse" mesh is then iteratively modified by adding one vertex after the other.

The interest of the meshing community for Delaunay triangulations arises from two principal facts: Firstly, the Delaunay triangulation provides a canonical way to connect the vertices through the empty sphere criterion (see Section 2.1.2 for some background on Delaunay triangulations); Secondly, the Bowyer-Watson algorithm [Bowyer, 1981, Watson, 1981] provides an efficient method to compute Delaunay triangulations incrementally in any dimension. This technique describes how to update efficiently a d-dimensional Delaunay triangulation when inserting a new vertex v. More precisely, each of the d-simplices of the triangulation whose circumscribing sphere contains the vertex v is removed from the triangulation, which creates a polygonal (in 2D) or a polyhedral (in 3D) cavity in the triangulation. The later is then updated by connecting each of the cavity vertices to v. In practice, the almost linear efficiency of the Delaunay mesh generation is a central advantage over the spatial decomposition and advancing front methods.

Early mesh generation algorithms based on the Delaunay triangulation start to generate the mesh vertices in a first stage and then they build a mesh in a second stage by computing the Delaunay triangulation of the set of generated vertices. Following this principle, the algorithm proposed by [Cavendish et al., 1985] starts to generate a grid of vertices out of slices of a three dimensional object, the resulting mesh is a sub-complex of the Delaunay triangulation of these vertices.

Although the Delaunay triangulations have been known for many years [Delaunay, 1934], it was not until the work of [Hermeline, 1982] and [Frey, 1987] that the Delaunay triangulation was utilized to guide the placement of vertices. Both articles lay the foundations of *Delaunay refinement* mesh generation methods. The idea boils down to, in a first stage, insert the vertices on the boundaries of the object to be meshed sufficiently close to each other so that the elements of the boundary of the object are represented in the Delaunay triangulation as a union of simplices. In a second stage, the simplices inside the object that do not satisfy the size or shape criteria are destroyed by inserting a point in their barycenter or at the center of their circumscribing sphere.

The mesh generation idea based on the Delaunay refinement paradigm has been reused later on so much that it became by far the most popular of the simplicial meshing techniques. In the late eighties, [Chew, 1989] introduced the first Delaunay refinement mesh generation algorithm with theoretical guarantees. This algorithm can mesh a two dimensional polygon, with a Delaunay triangulation, in a way that all three angles of any triangle inside the polygon are between 30°

and 120°. This guarantee comes however at a cost: the input polygon cannot have internal angles smaller than 90°. Moreover, this algorithm produces a uniform mesh, *i.e.* all triangles are more or less of the same size. A couple of years later, an extension of Chew's algorithm was proposed by [Ruppert, 1993] birthing a two dimensional meshing algorithm of planar straight line graphs (PSLGs), *i.e.* polygons with potential holes and pending edges. Ruppert's algorithm uses a Delaunay triangulation where the size of triangles may vary depending on a size criterion, so that all triangles angles are larger than $\approx 20.7°$. Ruppert showed also that the meshes generated by his algorithm are optimal in terms of size. Moreover, he obtains similar guarantees as the previously described hierarchical grid mesh generation algorithm of [Bern et al., 1990], still, in practice, Ruppert's algorithm generates meshes with fewer triangles. However, similarly to Chew's algorithm, Ruppert's algorithm cannot mesh planar linear graphs with inter-edges angles smaller than 60°.

This limitation is lifted by the *Corner Looping* algorithm proposed by [Bern et al., 1994] and by [Ruppert, 1995] were PSLGs are replaced by modified graphs that do not contain any small angles. A few years later, [Shewchuk, 1996, Shewchuk, 2000, Shewchuk, 2002] introduced the *Terminator* algorithm that can mesh planar linear graphs with small angles. To put it briefly, Shewchuk's algorithm generates meshes with an angle bound on triangles of up to 20.7° except for the angles of triangles in the vicinity of small input angles. Compared to the Corner Looping algorithm, the Terminator algorithm inserts less vertices around small angles. Later on Schewchuk extended his algorithm into three dimensions in [Shewchuk, 1998] were he proposes a method to mesh piecewise linear complexes, *i.e.* the nesting of polyhedra with possible pending facets. The generalization to three dimensions is relatively straightforward, albeit not without complications. Particularly, complications may arise in three dimensions when the algorithm is asked to mesh objects with boundaries forming angles smaller than 90°.

Based on idea proposed by [Cohen-Steiner et al., 2002], Cheng and Poon [Cheng & Poon, 2003] designed the first algorithm that places protecting balls on the incident edges to facets forming an angle smaller than 90°. This idea was later modified and made more practical by [Cheng et al., 2004, Cheng et al., 2007b, Cheng et al., 2007a].

In the early nineties, a smooth surface mesh generation algorithm based on the Delaunay refinement paradigm was designed by [Chew, 1993].This algorithm was proposed however without the theoretical study. More recently, [Boissonnat & Oudot, 2005] reprise the algorithm of Chew and studied its termination condition and the approximation guarantees provided by this algorithm. More details on Boissonnat and Oudot algorithm will be provided in Section 6.

5.5 A motive for further development

Meshes are commonly used in many applications to represent 3D shapes and are often generated by isosurfacing implicit representations obtained by surface reconstruction methods. However, if automated, the mesh generation process is prone to errors, resulting in meshes that are rarely fully satisfying. Practically, the mesh quality in term of vertex sampling and triangle quality can be improved. This process is also called *remeshing*. Remeshing is especially challenging when dealing with surfaces with sharp features. Recent approaches [Tournois et al., 2009, Yan et al., 2009] need the practitioner to manually tag the edges of an input mesh that correspond to features and then use some constrained optimization to sample them properly. More recently, [Lévy & Liu, 2010] designed an algorithm that uses an anisotropy field oriented along facet normals in a so called L_p *Centroidal Voronoi Tesselation* objective function to preserve the sharp features of the input mesh without any need to tag them manually. Also their method has the nice property of globally optimizing the vertices locations.

A large body of literature on the topics of automatic mesh generation and surface reconstruction (described extensively in Chapter 4) emerged over the past thirty years. However, what is noticeable, is that mesh generation and surface reconstruction were often tackled separately. To our knowledge, there have been only a handful of techniques that combined both [Kobbelt et al., 2001, Schreiner et al., 2006, Boissonnat & Oudot, 2005, Cheng et al., 2007b] along with their pros and cons, the challenge being to find a good balance between flexibility, robustness and accuracy. This is one research direction we got interested in, and that will be brought to you in more details in Chapter 7, leading to the following publication [Salman et al., 2010].

Part II

From 3D point clouds to meshes: Our Contributions

Nothing is more important than
to see the sources of invention
which are, in my opinion
more interesting than
the inventions themselves.

Gottfried Leibniz (1646-1716)

6

Smooth surface reconstruction

In this chapter we describe an original method to reconstruct a three dimensional scene from a sequence of calibrated images. Our approach uses both the dense 3D point cloud extracted by the multi-view passive stereo algorithm described in Section 3.2 and the calibrated images. It combines depth maps construction in the image planes with surface reconstruction through restricted Delaunay triangulation. The presented method may handle very large scale outdoor scenes. Also, its accuracy has been tested on numerous outdoor scenes including the multi-view benchmark proposed by [Strecha et al., 2008]. Our results show that the proposed method compares favorably with the current state of the art.

6.1 Introduction

6.1.1 Motivation

Various applications in Computer Vision, Computer Graphics and Computational Geometry require a surface reconstruction from a 3D point cloud extracted by stereovision from a sequence of overlapping images. These applications range from preservation of historical heritage to e-commerce through computer animation, movie production and virtual reality walkthroughs.

As described in our surface reconstruction survey in Chapter 4, early work on surface reconstruction has been focused on data acquired through laser range scanners, with characteristics that the obtained 3D point cloud is dense and well distributed [Ohtake et al., 2003a, Kazhdan et al., 2006, Hornung & Kobbelt, 2006, Mullen et al., 2010]. Although these reconstruction methods have been reported as successful, the nature of the laser scanners greatly limits their usefulness for large-scale outdoor reconstructions as reminded in Chapter 3. This is especially true when range finders need to be attached to flying balloons [Banno & Ikeuchi, 2005] to get, for instance, a detailed reconstruction of a large-scale outdoor scene such as the Bayon temple at Angkor in Cambodia [Banno et al., 2008].

The recent advances in multi-view passive stereo provide an exciting alternative for outdoor scenes reconstruction. However, the extracted 3D point clouds are entangled with redundancies, a large number of outliers and noise. Fortunately, on top of these point clouds, the additional information provided by the calibrated images can be exploited to help surface reconstruction.

In this chapter we will build on this idea and describe an original method to reconstruct a three dimensional scene from a sequence of calibrated images.

6.1.2 Related work

Over the past three decades there have been significant efforts in surface reconstruction from images, typically targeting urban environments or archaeological sites. This has led to the development of various algorithms whose results are getting significantly closer to the precision of surface reconstruction from point sets acquired from laser range scanners. It is arduous to bring up a classification of these works since many algorithms combine different approaches.

According to a recent survey provided by [Seitz et al., 2006], surface reconstruction methods can be roughly classified into four categories. The first category works by computing a cost function on a three dimensional volume, which is used later on to extract a surface mesh from it. It includes the voxel coloring algorithms [Seitz & Dyer, 2002] and the graph cut algorithms [Kolmogorov & Zabih, 2002, Vogiatzis et al., 2005, Lempitsky et al., 2006, Tran & Davis, 2006, Vogiatzis et al., 2007]. The second class includes the variational methods and work by iteratively evolving a surface to minimize an error function. This category includes methods based on level sets [Faugeras & Keriven, 1998a, Faugeras & Keriven, 1998b, Pons et al., 2007] which, as the space carving methods [Kutulakos & Seitz, 2000], starts from a large initial volume and shrinks it to fit the surface (for more details on level set methods please refer to Section 4.2.2). The third class of methods is based on the combination of multiple depth maps, ensuring a consistent 3D scene [Strecha et al., 2004, Gargallo & Sturm, 2005, Strecha et al., 2006, Merrell et al., 2007, Goesele et al., 2007, Zach et al., 2007, Pollefeys et al., 2008]. Finally, we call the fourth category of algorithms as hybrid algorithms as they combine multiple approaches. These hybrid methods usually proceed in two phases: First, they extract a set of interest points that are robustly matched across the images; second, they fit a surface to the reconstructed 3D points, possibly using the information coming from the images.

Among the presented classes our method falls in the last category. Techniques closest to ours generally start with a two dimensional Delaunay triangulation of the projections of the extracted 3D point clouds on the plane of one of the images, and then lift the triangulation into 3D space to obtain a surface model. In their work, [Pollefeys et al., 2004] use this two dimensional Delaunay triangulation and lifting approach on the interest points from one of the images, to create a mesh model. If the reconstructed 3D points are uniformly distributed, their Delaunay triangulation produces triangles of balanced sizes and shapes, suitable for polyhedral representation of a curved surface. Hence, this method works well for dense point samples, but for sparse point clouds, it often links points from different planar surfaces, leading to artifacts where some of the triangulation edges may not coincide with physical edges.

The concept referred to as *Image-Consistent Triangulation* in multi-view stereo reconstruction methods was first addressed by [Morris & Kanade, 2000]. They start by building an initial surface mesh based on lifting a two dimensional Delaunay triangulation of an acquired 3D point set of the scene. An edge swapping technique, is then used to traverse possible triangulated surfaces with vertices on the 3D positions of the matched interest points. They use a greedy approach to search for the triangulation whose connectivity is "photo"-consistent with the greatest possible number of views. In the same vein, [Nakatuji et al., 2005] detect texture discontinuities in images and swap the edges of the initial triangulation of a given 3D point set of the scene to match the detected discontinuities as much as possible. This results in a triangular mesh that is compatible with the physical object shape. Both approaches rely on statistical methods, such as simulated annealing, to find the optimal triangulation. These statistical methods are computationally expensive, and their convergence cannot be guaranteed. Therefore, both approaches [Morris & Kanade, 2000, Nakatuji et al., 2005] work well mainly with sparse input point clouds.

Hybrid approaches have been proposed by combining the calibrated images informations and

visibility constraints. The first algorithm addressing three dimensional scene reconstruction from sparse 3D data with arbitrary geometry and multiple occluding objects was presented by [Manessis et al., 2000]. A complimentary algorithm was proposed by [Hilton, 2005], he lifts in 3D space the two dimensional Delaunay triangulations in all views, and then rejects 3D triangles depending on the visibility constraints of all views. This algorithm is reliable and computationally efficient, the output is a set of triangulated surface patches consistent with all visibility constraints, but not a mesh.

Surface reconstruction from multiple views can also be thought in terms of the scenes they can handle. The requirements for the ability to handle large-scale scenes discards most of the multi-view stereo reconstruction algorithms including the algorithms listed in the Middlebury challenge [Seitz et al., 2006]. The methods which have proved to be more adapted to large-scale outdoor scenes are the ones that compute and merge multiple depth maps. However, in many situations, the reconstructed three dimensional models lack of accuracy or completeness due to the difficulty to consistently take visibility into account. Recently, three different accurate reconstruction algorithms have been proposed in [Furukawa & Ponce, 2007], [Labatut et al., 2007, Vu et al., 2009] and [Lafarge et al., 2010] to overcome this difficulty.

In their work, [Furukawa & Ponce, 2007] proposed a reconstruction method that generates and propagates a dense set of patches. This method relies on a careful succession of heuristics to filter and expand sets of oriented patches. The final surface is reconstructed through Poisson surface reconstruction of [Kazhdan et al., 2006] which does not handle any visibility issue. In the same year, [Labatut et al., 2007] designed an algorithm that mixes interestingly a three dimensional volumetric approach and an implicit method. They build a 3D Delaunay triangulation of their quasi-dense 3D point cloud. From that triangulation they reconstruct the scene object by labeling the tetrahedra as inside or outside the object. A globally optimal label assignment is efficiently found using graph cut optimization methods on the triangulation adjacency graph. Adjacency edges are weighted according to photo-consistency and the visibility constraints. Recently, Labatut *et al.* revamped each individual step of their approach and added a mesh-based variational refinement step in [Vu et al., 2009]. More recently, [Lafarge et al., 2010] interestingly mix a three dimensional mesh and geometric primitives, using a *Jump-Diffusion* process [Grenander & Miller, 1994], to output compact models while preserving details. To our knowledge, among the works described above, only [Vu et al., 2009, Lafarge et al., 2010] have shown a real foretaste of scalability and impressive results on large-scale outdoor scenes.

In this lineage of highly scalable approaches, [Jachiet et al., 2010] designed an algorithm for automatic simplified piecewise planar three dimensional reconstruction and completion from large 3D point sets that are entangled with noise and outliers. Their method has shown impressive practical results, but does not aim at detailed reconstructions, it rather focuses on reconstruction with strong shape priors to build geometrically *simplified* models from point clouds that can attain millions of points.

To demonstrate the efficiency of our reconstruction results we will rely on the publicly available quantitative challenge from the review of [Strecha et al., 2008] currently led by [Vu et al., 2009].

6.1.3 Contributions

Our reconstruction method is close to [Hilton, 2005] and to the *Image-Consistent Triangulation* concept proposed by [Morris & Kanade, 2000]. It consists in a pipeline that handles large-scale scenes while providing control over the density of the output mesh.

A triangular depth map is built in each image plane, respecting the main contours of the image.

The triangles of all the depth maps are then lifted in 3D, forming a 3D triangle soup. The similarity of the triangle soup with the surface of the scene is further enforced by combining both visibility and photo-consistency constraints. Lastly, a surface mesh is generated from the triangle soup using Delaunay refinement methods and the concept of restricted Delaunay triangulation. The output mesh can be refined until it meets some user-given size and quality criteria for the mesh facets.

The reminder of this chapter is organized as follows. Section 6.2 gives some background on the mesh generation technique needed in our approach. In Section 6.3, we describe in details the different steps of our algorithm. Section 6.5 discusses numerical experiments that demonstrate the potential of our approach for reconstructing cluttered scenes from real-world data, before we conclude in Section 6.6.

6.2 Background

In this section we give some background on the Delaunay refinement mesh generation algorithm of [Boissonnat & Oudot, 2005]. We will use this algorithm to obtain our output mesh from the generated triangle soup. Please refer to Chapter 2 for a description of the geometric concepts used in our method.

6.2.1 Delaunay refinement surface mesh generation

The Delaunay refinement surface meshing algorithm builds incrementally a sampling $\mathcal{P}$ on the surface $\mathcal{S}$ whose restricted Delaunay triangulation $D_{\mathcal{S}}(\mathcal{P})$ is a good approximation of the surface $\mathcal{S}$. Recall that if $\mathcal{P}$ is an ε-sample of $\mathcal{S}$ for a sufficiently small ε then $D_{\mathcal{S}}(\mathcal{P})$ is a good approximation of $\mathcal{S}$ in a topological and geometric sense [Amenta & Bern, 1998]. One drawback of the concept of ε-sample is the fact that it is hard to control if a sample is an ε-sample of a surface, and even harder to construct an ε-sample of a given surface. In their work [Boissonnat & Oudot, 2005] introduced the notion of *loose ε-sample* whose main advantage over ε-sample is that they are easier to check and construct, while remaining asymptotically identical to ε-samples. They also proved that if a loose ε-sample of $\mathcal{S}$, with $\varepsilon \leq 0.091$, can be constructed then $D_{\mathcal{S}}(\mathcal{P})$ will be a good approximation of $\mathcal{S}$.

Let $\mathcal{P}$ be a finite set of points and ε a real positive number. The point set $\mathcal{P}$ is called a loose ε-sample of $\mathcal{S}$ if the following conditions are satisfied:

1. $\mathcal{P} \subset \mathcal{S}$;
2. $D_{\mathcal{S}}(\mathcal{P})$ has vertices on all connected components of $\mathcal{S}$;
3. the center c of all the surface Delaunay balls of $\mathcal{P}$ is at a distance of $\mathcal{P}$ smaller or equal to $\varepsilon \operatorname{lfs}(c)$.

Because the surface $\mathcal{S}$ is unknown, the lfs is not known as it depends on the thickness and curvature of the object. Thus, the third condition is replaced by a distance criterion that adapts to local curvature and controls the approximation, and a size criterion which should take care of the thickness. The algorithm starts by sampling a small initial set of points $\mathcal{P}$ on $\mathcal{S}$. In practice, three points per connected component of $\mathcal{S}$ are sufficient. Then it computes the Delaunay triangulation $D(\mathcal{P})$ and its restriction $D_{\mathcal{S}}(\mathcal{P})$. This initial approximation is then refined until it meets the following quality criteria for facets of $D_{\mathcal{S}}(\mathcal{P})$:

- **minimum angle** α – This quality criterion ensures that the facets of $D_{\mathcal{S}}(\mathcal{P})$ are well shaped. The parameter α is the lower bound for the minimum angle of any mesh facet. The algorithm is guaranteed to end if $\alpha \leq 30°$.
- **maximum edge length** l – This sizing criterion is the upper bound for the maximum edge length of a facet of $D_{\mathcal{S}}(\mathcal{P})$.
- **maximum surface distance** d – This criterion governs the approximation accuracy of the mesh elements. The parameter d is the upper bound for the distance between the center of a facet of $D_{\mathcal{S}}(\mathcal{P})$ and the center of any surface Delaunay ball of this facet. This parameter has to be small compared to the aforementioned maximum edge length l to increase the approximation accuracy of the resulting mesh.

The refinement process tracks *bad facets* in $D_{\mathcal{S}}(\mathcal{P})$, *i.e.* facets that do not comply to the meshing criteria. Each bad facet is killed out of the mesh by inserting the center of its surface Delaunay ball. Such a Delaunay refinement meshing algorithm is greedy, it inserts points one after the other and maintains the current set of points P, $D(\mathcal{P})$ and $D_{\mathcal{S}}(\mathcal{P})$. It also maintains a list of bad facets. The refinement routine is iterated until there is no more bad facets left.

The key to [Boissonnat & Oudot, 2005] algorithm flexibility is it's ability to mesh surfaces as general as piecewise smooth surfaces. The only assumption that is made is that the input surface representation is amenable to simple geometric queries, namely intersection with a line segment. Put differently, the shape to be discretized is required to be known through an *oracle* that given a line segment detects whether the segment intersects the surface, and if so, returns the intersection points. Thus, the input shape can be represented as any surface reconstructed from a 3D point set, an implicit function or even as a triangle soup. Moreover, such a Delaunay refinement surface meshing algorithm yields a quality surface mesh, free of self-intersections.

6.3 Algorithm

Our algorithm consists in a pipeline that handles large-scale scenes while providing accurate reconstructions. The algorithm takes as input a sequence of calibrated images and a set of *tracks*, provided by the multi-view passive stereo technique described in Sections 3.2. Each input track stores its 3D point location as well as the list of camera locations from which it was observed during acquisition. The algorithm consists in three main steps:

1. merging, filtering and smoothing of the tracks;
2. using the pre-processed 3D points and the camera projection matrices we extract a triangle soup that minimizes visibility and photo-consistency violations;
3. computing from the triangle soup a mesh modeling the scene.

The different steps of our algorithm are illustrated in Figure 6.1. The following sections describe the three steps of the algorithm in more details.

6.3.1 Merging, filtering and smoothing

Given a set of calibrated images we first start by extracting a set of 3D points using the multi-view passive stereo technique of [Vu et al., 2009] described in Section 3.2. Figure 6.2 shows the typical input to our surface reconstruction algorithm provided by the point cloud extraction

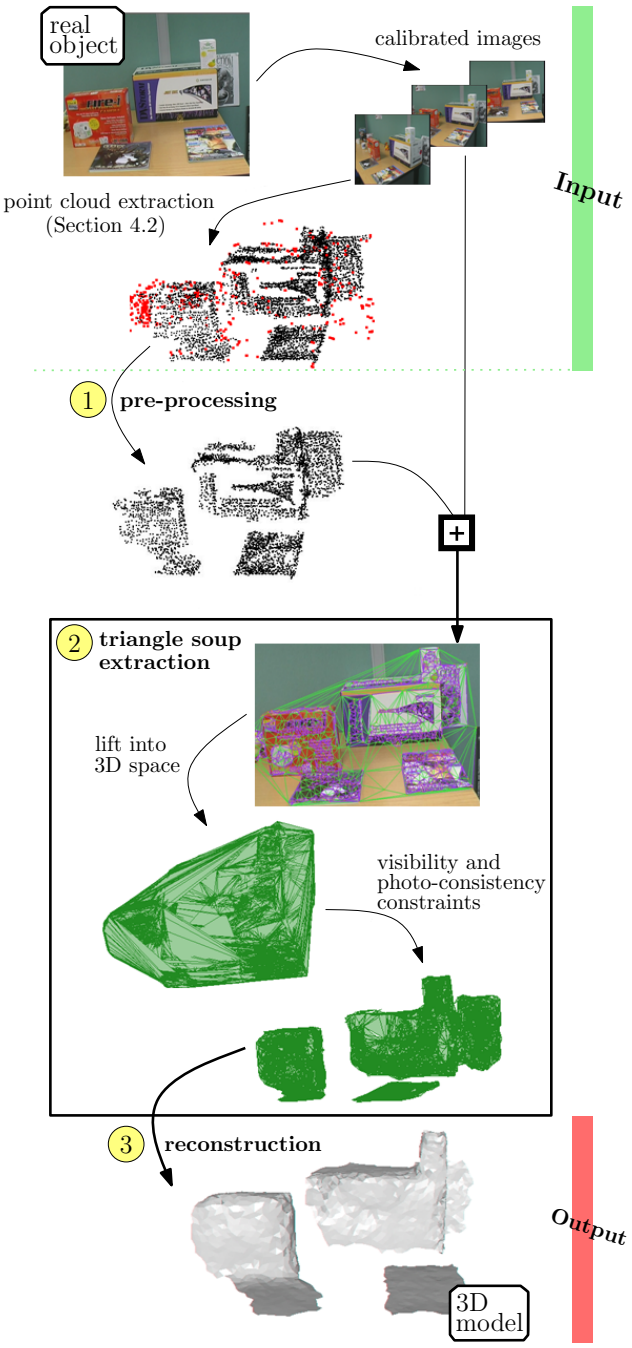


Figure 6.1: Reconstruction pipeline overview. Calibrated images are used by the multi-view passive stereo algorithm of [Vu et al., 2009] to extract a 3D point cloud. (1) We will first pre-process the 3D point cloud. (2) Using the pre-processed 3D points and the camera projection matrices we extract a triangle soup that minimizes visibility and photo-consistency violations. (3) Finally, we use the triangle soup as a surface approximation and build the output 3D model. Note that the triangles lying on the table were voluntary removed for readability.

algorithm. In this case the results were obtained from a set of twenty-five images taken from slightly different view points.

In the following, we call *line-of-sight* the segment from a camera location to the 3D position of a track seen by this camera. Despite the robust matching process used by the multi-view passive stereo algorithm, the extracted tracks require some pre-processing including merging, filtering and smoothing.

Note: The merging, filtering and smoothing routines described in this section are part of the Point set Processing package [Alliez et al., 2010b] of CGAL [cga, 2010], thus are publicly available and can be tested by the reader. Please refer to Appendix A for a more detailed overview of the functionalities provided by the point set processing package.

Merging – The first pre-processing step in our method consists in merging tracks whose positions are too close. We do that by building an incremental Delaunay triangulation of the 3D positions of the tracks. Each time a track 3D position is to be added, its nearest neighbor is requested. Depending on the distance between the 3D positions of these two tracks, one of two cases can arise:

- if the nearest neighbor is close, with respect to a threshold t_1, the two tracks are merged in a single one and the list of cameras is updated to include the union of the cameras of the two merged tracks;
- if the closest neighbor is at a distance larger than t_1 then both tracks are preserved in the final set of tracks.

Filtering – As tracks are coming from characteristic points detected in the images, their 3D locations should be densely spread over the surface of the scene objects. Thus, tracks which are rather isolated are likely to be outliers. Another criterion used to further filter out outliers, is based upon the angle between lines-of-sight. When a 3D point location is computed from lines-of-sight forming small angles, the intersection computation is imprecise and the 3D point is likely to be an outlier. For these reasons we use two criteria to detect outliers in the set of tracks: the *distance to neighbors* and the *cone angle*.

- **Distance to neighbors -** This criterion aims at eliminating tracks far away from densely populated regions of space. We compute for each track with 3D position p_i the average squared Euclidean distance $d_k(p_i)$ from p_i to its k-nearest neighbors.

$$d_k(p_i) = \frac{1}{k} \sum_{p_j \in \mathrm{kNN}(p_i)} \|p_i - p_j\|^2 \ .$$

  In practice, for the 3D point clouds of the scenes we wish to reconstruct, the distribution of the average distance to the k-nearest neighbors tends to be Gaussian-like. Thus, if σ is the standard deviation of the distance distribution, we eliminate tracks with distance higher than 3σ.

- **Cone angle filtering -** This criterion aims at eliminating the tracks which have been observed from only a few camera locations or from a set of cameras in directions forming small angles from the 3D point. We compute for each track with 3D position p_i the aperture angle of the smallest cone with apex at p_i containing all cameras that observe point p_i. Tracks with cone angles smaller than a threshold t_2 are discarded. Note that this criterion is very useful for filtering 3D point clouds extracted from a set of images taken from a video sequence.

Smoothing – The final pre-processing step smooths the remaining tracks using a smoothing algorithm based on a polynomial fitting of osculating jets and reprojection [Cazals & Pouget, 2003]. In contrast to [Cazals & Pouget, 2003], we use Gaussian weights for the optimization steps. Let p_0 be the 3D position of a track and $\text{kNN}(p_0) = \{p_1, \ldots, p_k\}$ its k-nearest neighbors. A principal component analysis (PCA) of this set of nearest points provides a first order estimate of the tangential plane at p_0 and the reference domain for the polynomial fitting. The estimated normal $\overrightarrow{n_0}$ of this tangent plane at the point p_0 is the solution of the following optimization problem:

$$\min_{\|\overrightarrow{n}\|=1} \sum_{m=1}^{k} \langle \overrightarrow{n}, (p_0 - p_m) \rangle^2 \gamma(\|p_0 - p_m\|) \,,$$

where $\gamma(x) = e^{-\frac{x^2}{2}}$ is a Gaussian weight function giving more importance to points close to p_0.

This estimated tangent plane is only used as the reference domain to compute a bivariate polynomial approximation to the neighboring points $\text{kNN}(p_0)$ with the estimated tangent plane as reference domain. Using the weights $h_m = \gamma(\|p_0 - p_m\|)$, the paraboloid $\pi(x, y) = ax^2 + by^2 + cxy + dx + ey + f$ minimizing

$$\min_{\pi \in \Pi_2^2} \sum_{m=1}^{k} (\pi(x_m, y_m) - z_m)^2 h_m \,,$$

where Π_2^2 denotes the bivariate polynomials of second degree, can be found by solving the following linear system of equations

$$A^T \begin{pmatrix} h_1 & & 0 \\ & \ddots & \\ 0 & & h_k \end{pmatrix} A \begin{pmatrix} a \\ b \\ \vdots \\ f \end{pmatrix} = A^T \begin{pmatrix} h_1 z_1 \\ h_2 z_2 \\ \vdots \\ h_k z_k \end{pmatrix} ,$$

with

$$A = \begin{pmatrix} x_1^2 & y_1^2 & x_1 y_1 & x_1 & y_1 & 1 \\ & & \vdots & & & \\ x_k^2 & y_k^2 & x_k y_k & x_k & y_k & 1 \end{pmatrix} .$$

Practically, for each track 3D position, we compute its corresponding low degree polynomial surface that approximates its k-nearest neighbors as aforementioned. We then project the track 3D position onto the computed polynomial surface following the normal to the graph of the polynomial at the origin $(0, 0)$: $\overrightarrow{n_0} = (-d, -e, 1)^T$. In the presence of noise, this normal approximates much better the surface normal than the previously computed first order estimate of the tangent plane.

Note: For all the presented results we used $t_1 = 0.001\beta$, where β is half the diagonal of the bounding box of the point cloud, $t_2 = 0.08$radian, and k equals to 150 and 85 respectively for the filtering and smoothing step.

6.3.2 Triangle soup extraction

The next step in our method combines the pre-processed tracks and the input images to build in each image plane a triangular depth map that comply with the image discontinuities. The

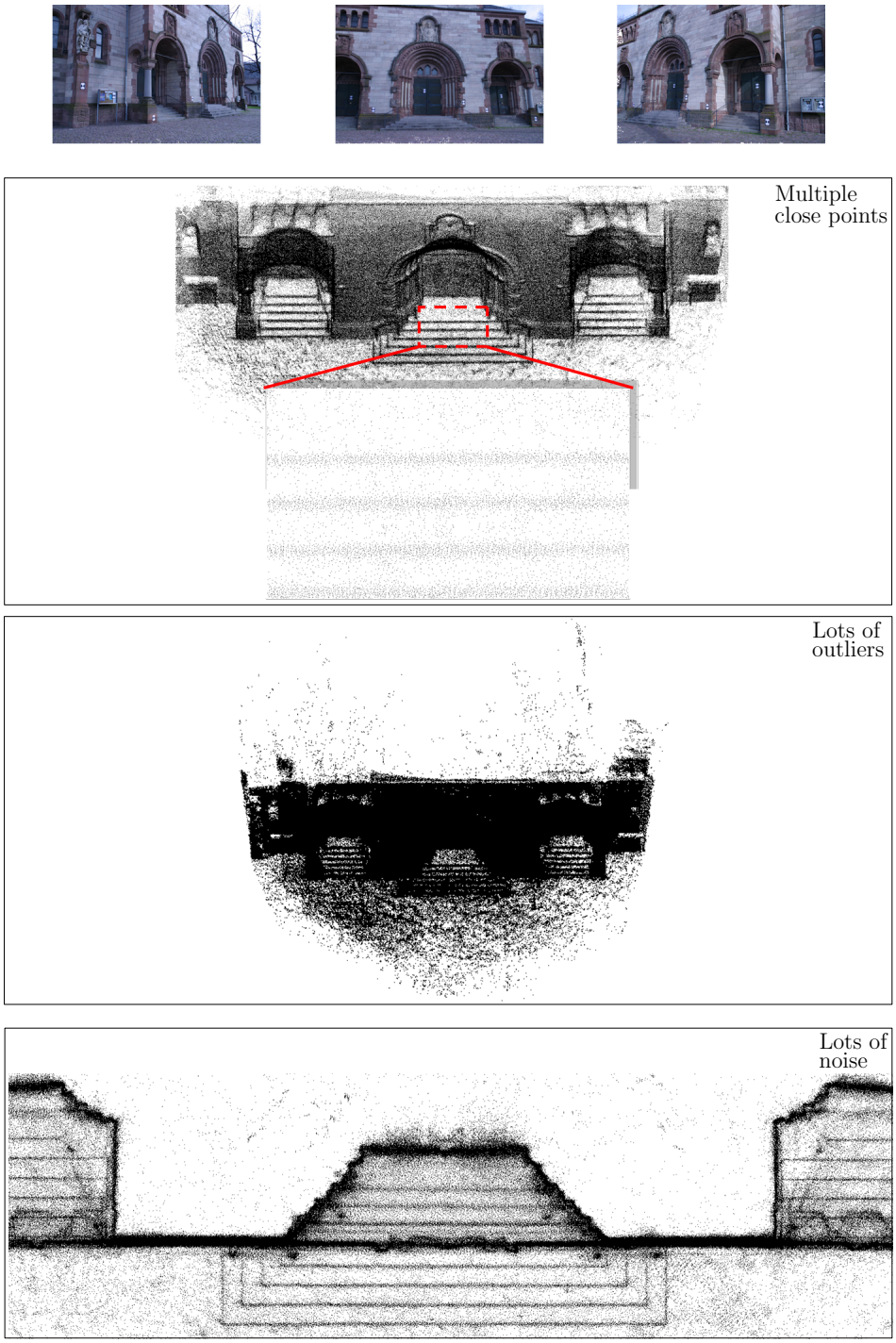


Figure 6.2: Top to bottom: three images out of twenty-five of the *Herz-Jesu-p25* dataset of [Strecha et al., 2008]; oblique view of the extracted point cloud, note how points are close to one another; un-zoomed view, note the high number of outliers; top view note the high amount of noise.

triangles of each depth map are then lifted into 3D space forming a triangle soup which will require further filtering to minimize violations of visibility and photo-consistency constraints.

Depth Maps Construction – Since straight or curved edge segments appearing in the images are one of the most important keys for understanding or reconstructing the scene from two dimensional images, it is expedient to include this edge information in the depth maps construction process.

A typical *edge* is a set of points where the brightness intensity of the corresponding pixels changes most strongly in their neighborhood. Thus, on a two dimensional image the edge corresponds to a contour and often a sharp edge or an occluding boundary in 3D space.

A simple yet powerful method for edge detection is to perform a contrast analysis into the input images using gradient-based techniques. To that aim, we used the standard Canny edge detection algorithm [Canny, 1986], that has the following properties:

- high positional accuracy of an edge, even with aliasing;
- good sensitivity to high frequency data;
- reduction of the data (many pixels dont produce edge points).

The last of these properties, *i.e.* reduction of the data, is important as it can make a significant difference in the computation time of the depth maps.

First, the input image I is convolved with a Gaussian kernel $G(x, y) = e^{-(x^2+y^2)/2\sigma^2}$ of standard deviation σ (typically $\sigma = 2$), to give a scale-space representation $S = G * I$.

Then the gradient of the image is computed by feeding the smoothed image S to a convolution operation with the derivative of a Gaussian mask in both the vertical and horizontal directions. Note that the Gaussian mask and its derivative are separable, allowing the two dimensional convolution operations to be simplified. Practically, we compute the gradient of S using the centered three-by-three first difference Sobel filters, to produce two images S_x and S_y respectively for the x and y partial derivatives:

$$S_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} * S \quad \text{and} \quad S_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} * S \,.$$

From these gradients, the gradient magnitude M and the gradient direction Θ can be determined:

$$M = \sqrt{S_x^2 + S_y^2} \,.$$

$$\Theta = \arctan\left(S_y / S_x\right),$$

The magnitude image M will contain large values where image gradients are strong. However, when using the Sobel filters the edges can be very thick or narrow depending on how much the image was smoothed. Thus, to identify the edges properly, the multi-pixel wide edges should be thinned down to single pixel width. This is done using the *non-maxima suppression* technique which has the effect of suppressing all values along the direction of the gradient that are not peak values. The Canny algorithm starts by rounding the angle of the gradient direction Θ to one of four angles representing the vertical, horizontal and two diagonals.

Practically, a three-by-three neighborhood window is applied to the magnitude image M. At each pixel, the center element of M in the window is compared to its two neighbors along the gradient direction given by the angle value. If the magnitude value of the center element is smaller than one of its two neighbors, then it should not be classified as an edge element and

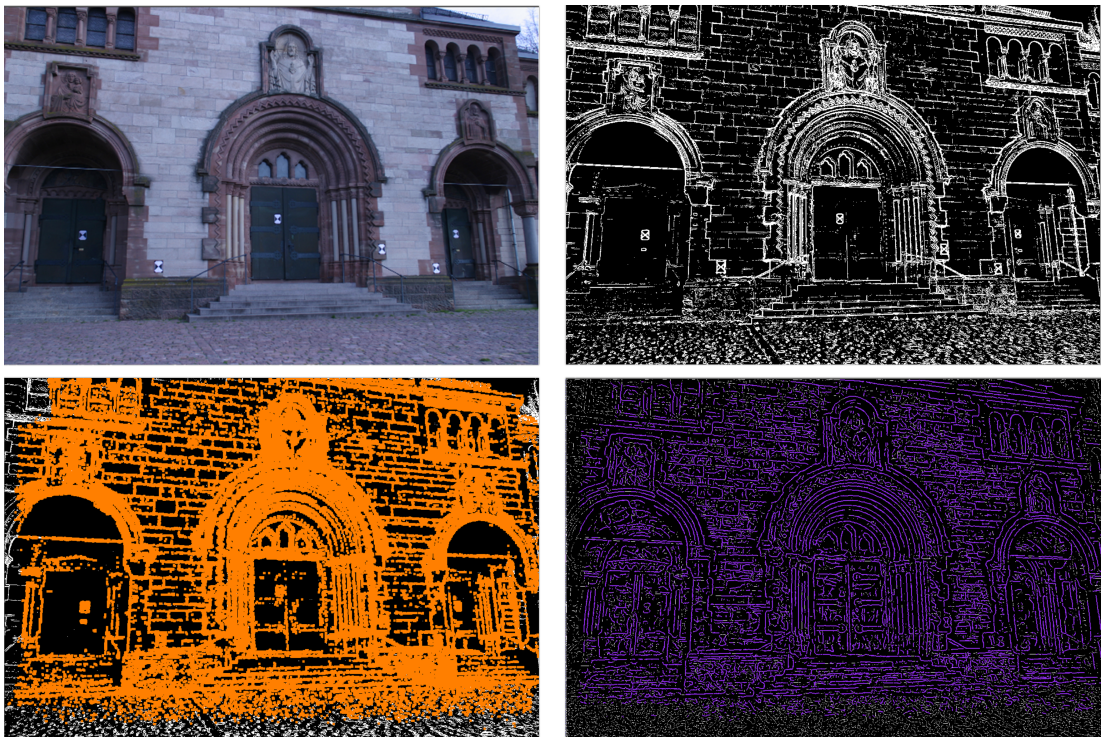


Figure 6.3: Top to bottom, left to right: one image out of twenty-five of the *Herz-Jesu-p25* dataset of [Strecha et al., 2008]; the extracted edges using the Canny edge detector [Canny, 1986]; set of ctracks whose projection is close to an edge (orange); approximation of the detected contours as a set of non-intersecting polylines (purple).

its magnitude value is set to zero . From this non-maxima suppression stage, a set of edges are identified and are retained in the form of a non-maxima suppressed image N.

Finally, instead of using a single threshold value for the image N, the Canny algorithm introduced *hysteresis thresholding* which takes N and applies two thresholds θ_1 and θ_2, where $\theta_1 > \theta_2$. When a pixel magnitude in N is greater than θ_1 it is recorded as *definite edge*, when a pixel magnitude is under θ_2 it is recorded as *non edge*, and when a pixel magnitude is between θ_2 and θ_1 it is recorded as *candidate edge*. Next, all candidate edge that can be traced back to a definite edge via adjacent candidate edges are also marked as definite edges. This process mollifies problems associated with edge discontinuities by identifying strong edges, while preserving comparatively weaker edges, and while maintaining some level of noise reduction. Figure 6.3 illustrates a typical output of this edge detection procedure.

We then compute, in each resulting binary image, an approximation of the detected contours as a set of non-intersecting polygonal lines with vertices on the projection of the tracks. For this we consider in each image the projection of the tracks located in this image. We call contour track, or *ctrack*, a track whose projection on the image is close to some detected contour. We call a non-contour track, or *nctrack*, any other track. We consider the complete graph over all ctracks of an image and select out of this graph, the edges which lay over detected contours.

The selection is performed by a simple weighting scheme adapted from the field of Computational Biology [Needleman & Wunsch, 1970]. We traverse every edge of the graph sequentially, if the edge passes through a contour pixel, it is a *match*, and we give the edge a reward score

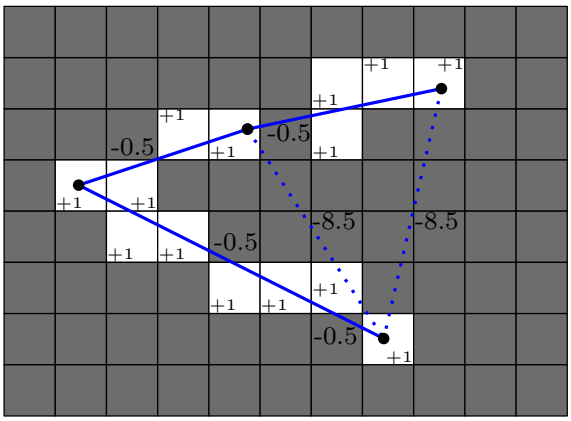


Figure 6.4: Weighting scheme: Delaunay triangulation of ctracks (black dots) superimposed to a binary image, only edges with a positive weight (straight blue line) are considered as an approximation of the detected contours.

of +1. Empirically, it is more likely that the edge does not lay on the contour if it crosses a big gap of length L. Said differently, it is more likely that an edge crossing L adjacent non-contour pixels is not part of the contour, than an edge crossing L small gaps of length 1. Therefore, we will use an *affine gap penalty* that will favor single gaps over longer gaps of the same total length. We will use a gap *opening* penalty $o > 0$ and a gap *extension* penalty $e > 0$ where $e > o$ (typically $e = 2$ and $o = 0.5$), to encourage gap introduction rather than gap extension. A gap of length L is then given a penalty $p = o + (L - 1)e$. The edge weight is then computed as the sum of match scores and gap penalties and only edges with a positive weight are kept. A sketch of the polygonal line approximation of the contours is illustrated in Figure 6.4.

Inherently, whenever there is n ctracks aligned on an image contour, we select the $n(n-1)/2$ edges joining these tracks. Albeit, we wish to keep only the $n-1$ edges between consecutive ctracks on the contours. For this reason we use a length criterion to further select a subset of non-overlapping edges fashioning the polygonal approximation of the contour.

In practice, in order to speed up the computation, we chose to consider the Delaunay triangulation over all ctracks instead of building a complete graph. Other steps of the algorithm remain unchanged were Delaunay edges are considered instead of the complete graph edges. This gives a good approximation because of the high density of ctracks per image. Figure 6.3 illustrates the output of this step.

In each image plane, edges in the polygonal contour approximation are used as input segments to a two dimensional constrained Delaunay triangulation with all projected tracks, ctracks and nctracks, as vertices. The output is a triangular depth map per image. The triangles of this depth map are then lifted in 3D space using the 3D coordinates computed by the multi-view passive stereo algorithm. However, lifted depth maps from consecutive images partially overlap one another and the triangle soup includes redundancies. When two triangles share the same vertices, only one is kept.

Triangle soup filtering – The vast majority –but not all– triangles of the soup lie on the actual surface of the scene. However, at occlusion boundaries, the depth map construction step might produce triangles that connect foreground surfaces to background surfaces. In the sequel, we note $\Gamma = \{\tau_0, \ldots, \tau_n\}$ the 3D triangle soup. We wish to filter Γ to remove erroneous triangles. Operationally, we proceed as follows:

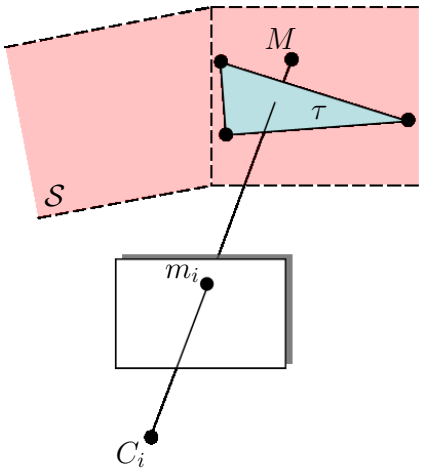


Figure 6.5: Visibility constraints: the point M seen in the current view through m_i has its free space violated by triangle τ seen in another view.

1. The *visibility constraint* states that the line-of-sight from a camera position to the 3D position of a track seen by this camera belongs to the empty space, *i.e.* it does not intersect any object in the scene. Therefore a triangle intersected by some line-of-sight should be removed from Γ (see Figure 6.5). However, this rule has to be slightly softened to take into account the smoothing step described in Section 6.3.1, where the tracks 3D positions have been slightly changed. More specifically, each triangle τ_i in Γ gets a weight w_i equal to the number of lines-of-sight intersecting τ_i. We then discard from Γ any triangle with a weight w_i greater than a threshold t_3. Practically, we insert all elements of Γ in an axis aligned bounding box tree (AABB tree) data structure, in order to speed up the triangle-ray intersection detection queries.

2. We also filter out any triangle whose vertices, at least one of them, are only seen by grazing lines-of-sight. In practice, a triangle is kept if the angle between the line-of-sight and the triangles normal does not exceed a given threshold (see Figure 6.6), typically 60 degrees is used in our experiments to characterize a grazing line-of-sight.

3. Depending on the density of the pre-processed tracks, the triangle soup can contain"big" triangles, *i.e.* triangles whose circumradii are big compared to the scale of the scene. We use *shape* and *photo-consistency* to filter big triangles.
   A triangle is said to be *anamorphic* if it has a too big radius-edge ratio, where the radius-edge ratio of a triangle is the ratio between the radius of its circumcircle and the length of its shortest edge. Anamorphic big triangles are likely to lie in free space and should be discarded. Big triangles with regular shapes are filtered using a photo-consistency criteria:
   A 3D triangle τ_i in Γ is photo-consistent, if its projections into all images where τ_i vertices are visible, *i.e. consistent images*, correspond to image regions with the same "texture". More precisely, among these consistent images, the one where the projected triangle presents the largest area is chosen as the reference image. After computing the homographies between the reference image and all other consistent images [Chen & Medioni, 1997], all triangles in the consistent images are warped onto the reference image and compared to it through NCC (see Section 3.2.2 for details on NCC). The NCC values are in the range of -1 to 1 , with 1 indicating a perfect correlation between triangles (exact match). A value of -1 indicates an inverse correlation and values near zero indicates a false match.

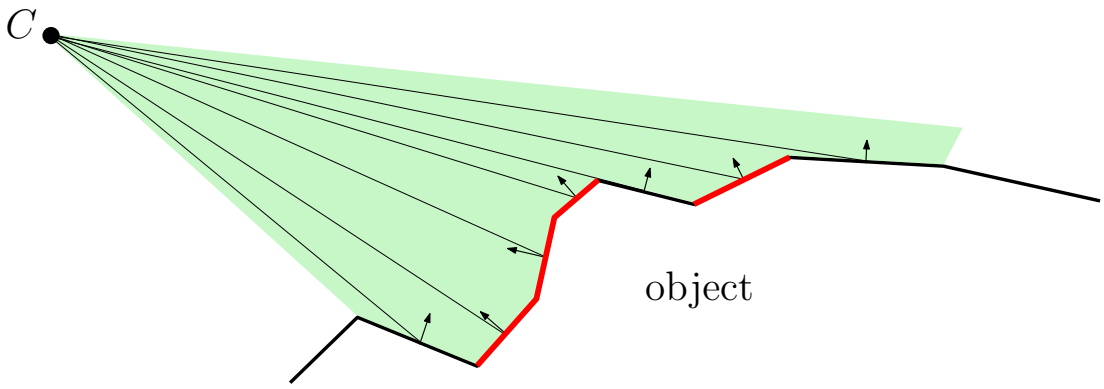


Figure 6.6: Cross-section of the triangle soup and camera viewing cone (green). Lines-of-sight are cast from the camera center C to each triangle that is seen by this camera. A triangle is kept (red) if the angle between the line-of-sight and the triangles normal does not exceed a given threshold.

We compute the average NCC value and compare it to a fixed threshold. The threshold value we use is 0.7, thus if the NCC value exceeds this threshold, the projected triangles are assumed to match and τ_i is preserved. In our algorithm, due to the high density of the input 3D point clouds, triangles lying near the surface are very small thus the photo-consistency term becomes almost useless. In practice, we skip this step however for sparse point clouds we would use NCC because it is known to be less sensitive to illumination changes between the views. Also, we would only use consistent images where the angle between the normal of τ_i and the line-of-sight through the center of the image is small. This scheme gives head-on view more importance than oblique views.

6.3.3 Reconstruction

For the final step, we use the Delaunay refinement surface mesh generation algorithm designed by [Boissonnat & Oudot, 2005] and described in Section 6.2.

Recall that this algorithm needs to know the surface to be meshed only through an oracle that, given a line segment, detects whether the segment intersects the surface and, if so, returns the intersection points. Since the triangles of Γ minimize violations of visibility and photo-consistency constraints, they can be used as an approximation of the surfaces we wish to model. We implemented the oracle required by the meshing algorithm using the triangles of Γ as an approximation of the surface. By doing so, the oracle will test and compute the intersections with Γ to answer to the algorithm queries. Technically, we use the updated AABB tree data structure from the filtering stage, to perform the intersection test efficiently.

In other words, to produce meshes with different resolutions, we just have to apply the reconstruction step on the extracted triangle soup and alter the Delaunay refinement parameters (α, l, d) (see Figure 6.7).

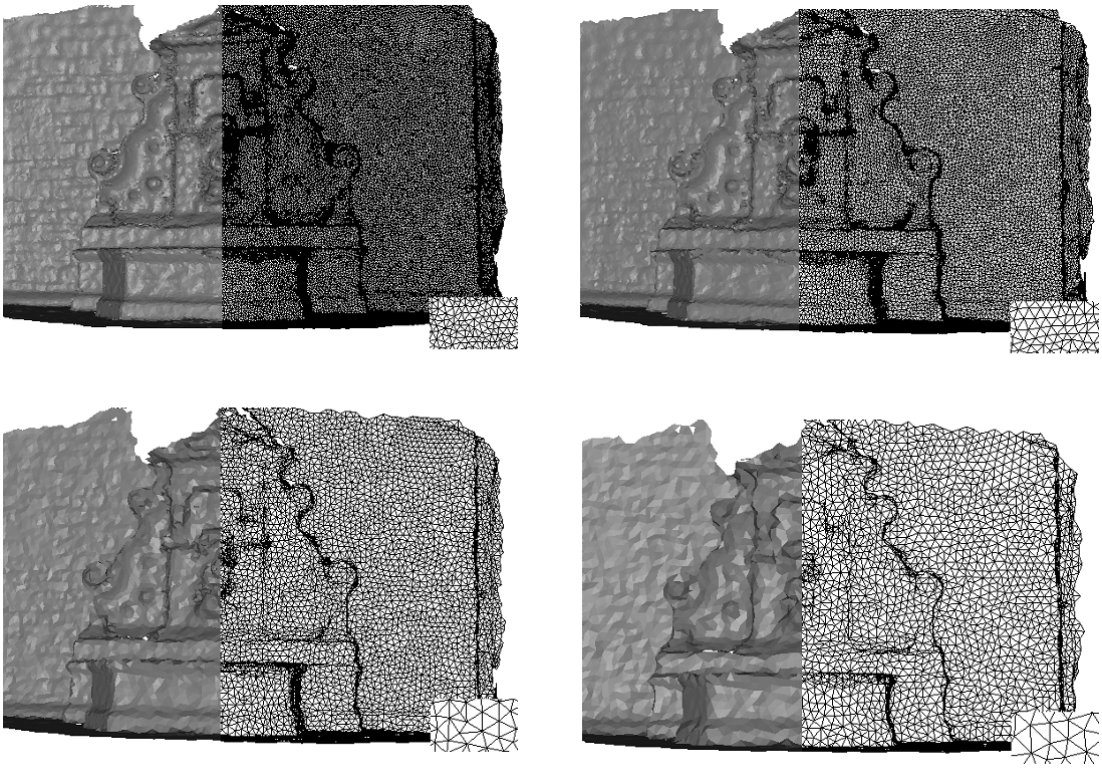


Figure 6.7: *Fountain-p11* dataset of [Strecha et al., 2008]. Top to bottom, left to right: 3D meshes resulting from our reconstruction step. By altering the parameters (α, l, d) in the reconstruction step, the output mesh becomes coarser, but, the fountain shape remains.

6.4 Implementation

We have implemented our approach in **C++**, using both CImg and Cgal libraries. The Cgal library, *i.e.* Computational Geometry Algorithms Library [cga, 2010], defines all the needed geometric primitives and provides very efficient algorithms to compute Delaunay triangulations in both two and three dimensions [Boissonnat et al., 2000]. More specifically, the implementation is robust to degenerate configurations and floating-point error, through exact geometric predicates, while being able to process millions of points per minute on a standard workstation.

The Cgal library also provides the point set processing package [Alliez et al., 2010b], among others, needed in the pre-processing step of our algorithm and that will be described in more details in Appendix A. Moreover, it provides access to the AABB tree data structure [Alliez et al., 2010c] which is an efficient data structure for intersection detection queries. More importantly it provides us with an implementation of the Delaunay refinement surface meshing algorithm [Rineau & Yvinec, 2010] that we use in the last step of our algorithm.

The CImg library, *i.e.* C++ Template Image Processing Toolkit [cim, 2010], is a toolkit for image processing that defines classes and methods to manipulate generic images. We use it for the edge detection, the constraint retrieval and the photo-consistency filtering step.

The oracle we implemented to test and compute the intersections with Γ to answer to the

Delaunay refinement surface mesh generation algorithm queries is defined by the following code.

```
#include <boost/static_warning.hpp>
#include <utility>
#include <CGAL/iterator.h>
#include <CGAL/AABB_tree.h>
#include <CGAL/AABB_triangle_primitive.h>
#include <CGAL/AABB_traits.h>

template <class K, class TriangleIterator>   // Value type Triangle_3
class AABB_trianglesoup_oracle
{
  typedef AABB_trianglesoup_oracle  Self;

public:
  typedef typename K::Point_3   Point_3;

  typedef Point_3         Intersection_point;
  typedef Self            Surface_3;

  typedef CGAL::AABB_triangle_primitive<K,TriangleIterator> AABB_prim;
  typedef class CGAL::AABB_traits<K,AABB_prim>        AABB_traits;
  typedef CGAL::AABB_tree<AABB_traits>            Tree;

private:
  Tree *m_pTree;

public:
  Tree* tree() const{ return m_pTree; }

  // constructors
  AABB_trianglesoup_oracle(){ m_pTree = NULL; }
  AABB_trianglesoup_oracle(Tree *pTree) { m_pTree = pTree; }

  class Intersect_3
  {
  const Self& self;

  public:
  typedef typename Tree::Object_and_primitive_id    Obj_prim_id;

  public:
  Intersect_3(const Self& self) : self(self){}

  template<class SegmentType>
  CGAL::Object operator()(const Surface_3& surface,
              const SegmentType& segment) const
  {
    typedef boost::optional<Obj_prim_id> Object_and_primitive_id;

    Intersection_point          intersect_pt;
        std::vector<Obj_prim_id>      obj_prim_vect;
    std::vector<Point_3>        point_vector;
    surface.tree()->all_intersections(segment,
                      std::back_inserter(obj_prim_vect));
```

```
54
55          std::vector<Obj_prim_id>::iterator it = obj_prim_vect.begin();
56
57          for(;it!=obj_prim_vect.end(); ++it)
58          {
59            const Point_3* p = CGAL::object_cast<Point_3>(&(it->first));
60            if(p)
61              point_vector.push_back(*p);
62          }
63          if(point_vector.empty())
64            return CGAL::Object();
65          else
66            return CGAL::make_object(CGAL::centroid(point_vector.begin(),
67                     point_vector.end()));
68        }
69    }; // end class Intersect_3
70
71    Intersect_3 intersect_3_object() const { return Intersect_3(*this); }
72
73  }; // end class AABB_trianglesoup_oracle
```

6.5 Results

In our experiments, we used for the point cloud pre-processing step the values of parameters $(t_1; t_2; k)$, as listed at the end of Section 6.3.1. Moreover, the visibility constraint threshold t_3 , defined at the end of Section 6.3.2, is fixed at 5 rays. Leaving aside the time required by the multi-view passive stereo track extraction step, in our experiments the runtime of the reconstruction ranges from 9 minutes for 200 000 tracks on 8 images to 38 minutes for 700 000 tracks on 25 images, on an IntelQuad Xeon 3.0GHz PC.

Strecha et al. [Strecha et al., 2008] provide quantitative evaluations for six outdoor scene datasets. To our knowledge these are the only available datasets and evaluation benchmark that allows comparison on large scale scenes. The evaluation of the reconstructed models is quantified through the cumulative error distribution, showing, for each deviation σ, the percentage of the scene recovered within an error less than σ. The deviation is estimated with respect to the *ground truth model* acquired with a LIDAR system (please refer to Section 3.1 for more details on the acquisition process used in [Strecha et al., 2008]).

We have tested our approach on the *fountain-p11*, *entry-p10*, *castle-p19* and *Herz-Jesu-p25* particularly challenging datasets. Focusing on the *fountain-p11* dataset, Figure 6.8 shows, two of the images, the extracted triangle soup, our reconstruction and the cumulative error distribution compared with the distributions obtained by other reconstructed methods. The multi-view passive stereo cloud has 392 000 tracks from which 35 percent were eliminated at the pre-processing step. Our algorithm has successfully reconstructed various surface features such as regions of high curvature and shallow details while maintaining a small number of triangles (500k). The comparison with other methods; VU [Vu et al., 2009], FUR [Furukawa & Ponce, 2007], ST4 [Strecha et al., 2004], ST6 [Strecha et al., 2006], ZAH [Zaharescu et al., 2007] and TYL [Tylecek & Sara, 2009] illustrates the effectiveness and flexibility of our method as it generates small meshes that are accurate and complete while being visually acceptable. Please note that we do not employ any post-processing variational refinement as in [Vu et al., 2009], that might be an interesting step to explore to get more accurate output meshes.

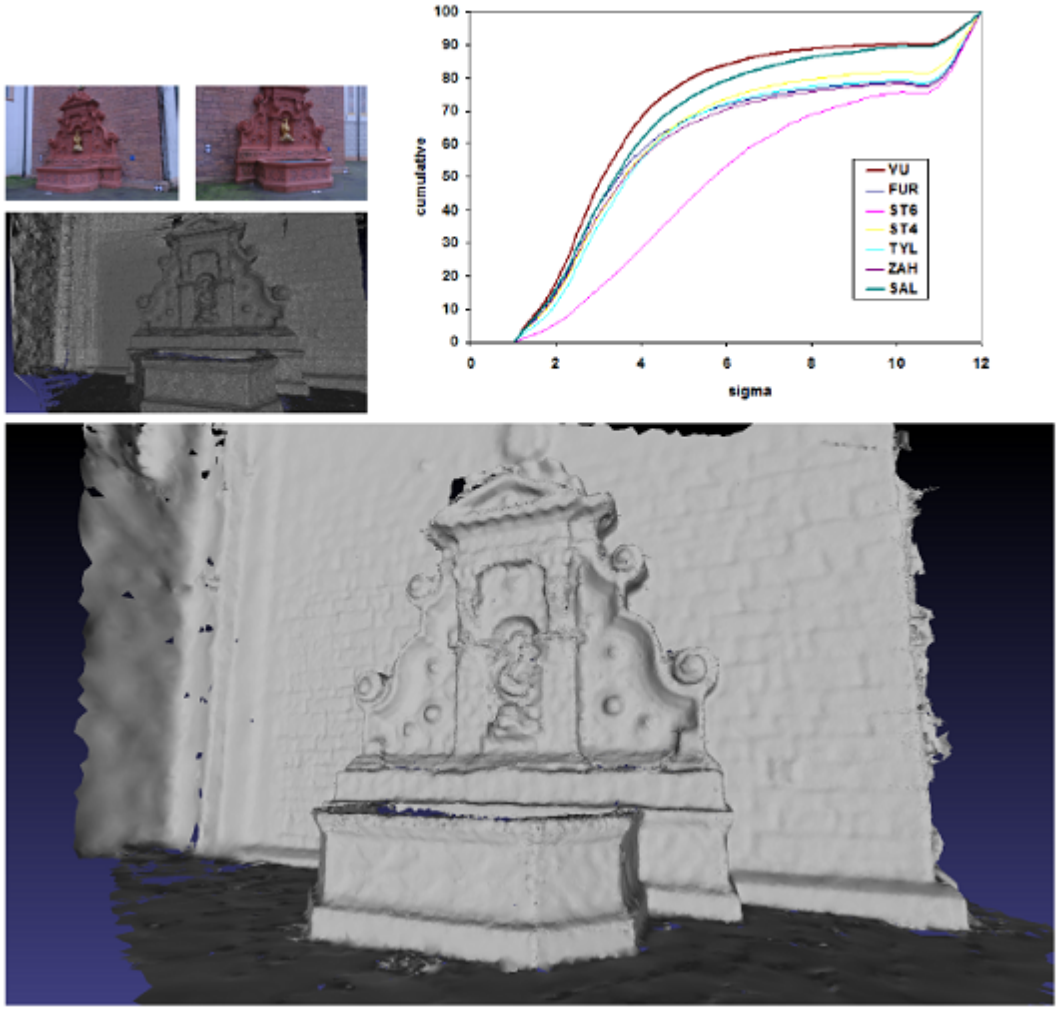


Figure 6.8: Top-left: two images of *fountain-p11* dataset of [Strecha et al., 2008] and the extracted triangle soup. Bottom: our reconstruction (500 000 triangles). Top-right: the cumulative error distribution (SAL for our work) compared with others. For comments please refer to main text.

Moreover, Figure 6.9 illustrates the ability of our algorithm to take into account user-defined size and quality criteria for the mesh facets on the *Herz-Jesu-p25* dataset. Figure 6.10 shows our reconstruction results for the *entry-p10* and the *Herz-Jesu-p25* datasets. For extended results we refer the reader to the challenge website [Strecha et al., 2008].

To show the ability of our algorithm to cope with much bigger outdoor scenes and its ability to produce meshes that take into account the user budget for the size of the output mesh, we have chosen to test our method on an aerial acquisition of the Aiguille du Midi mountain. This dataset also refered to as the *Aiguille du midi dataset* was graciously provided to us by Jean-Philippe Pons[1] and the copyright goes to Bernard Valet [2] for data acquisition and

[1] http://certis.enpc.fr/~pons
[2] http://www.bvallet.com

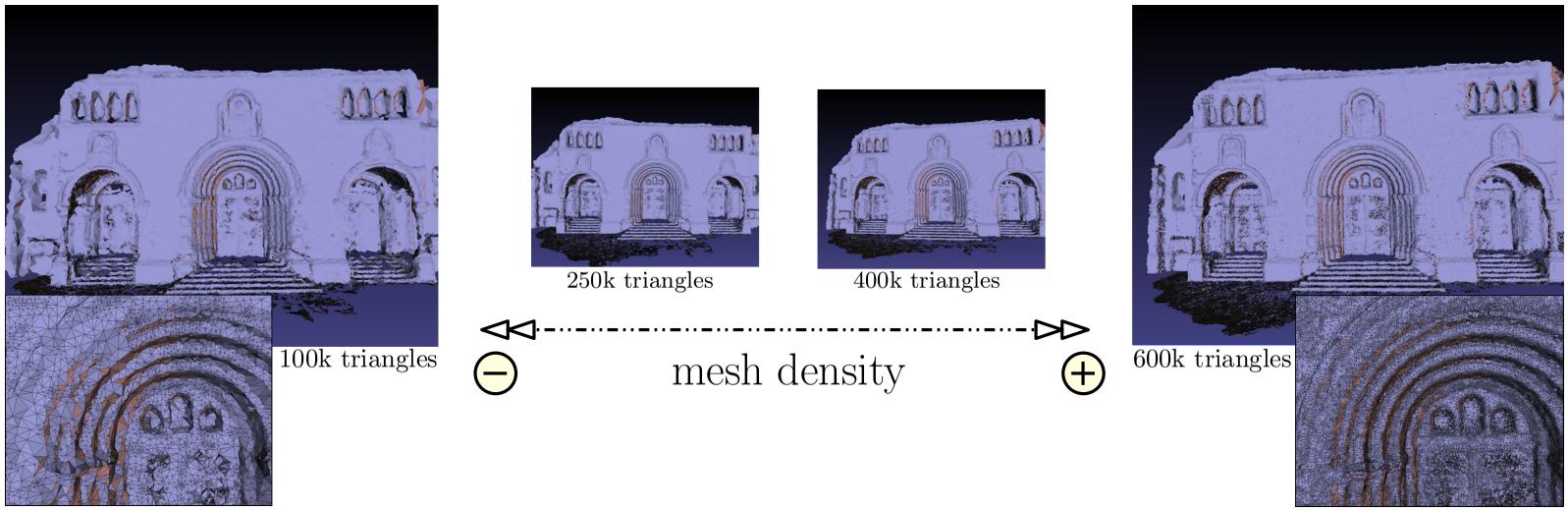


Figure 6.9: *Herz-Jesu-p25* dataset of [Strecha et al., 2008]. By altering the size and quality criteria for the mesh facets one can control the density of the output mesh.

calibration and to the CERTIS [3]. It includes fifty-one images, the extracted cloud has 6 700 000 points with lots of outliers. At the pre-processing step, eighty percent of the points were eliminated. The reconstructions shown in Fig. 8 are obtained with α equals 20 degrees and the couple (d, l) is set as (0.01, 0.035) and (0.002, 0.01) in unit of the bounding box half diagonal. This enabled us to reconstruct from a single extracted triangle soup output meshes of various resolutions (270 000 triangles and 1 000 000 triangles respectively in Figure 6.11). Note that details such as antennas and bridges are recovered even in the coarsest model. Table 6.1 shows the number of 3D points, the number of triangles for each reconstructed mesh and the time for the reconstruction algorithm.

6.6 Conclusion

In this chapter, we have presented a surface reconstruction algorithm from stereo vision data. Our algorithm copes with the fuzziness of the data by using information from both the multi-view passive stereo extracted 3D point clouds and the calibrated images. After filtering and smoothing the point cloud, the algorithm builds for each calibrated image a triangular depth map respecting the main contours of the image. The triangles of all depth maps are then lifted in 3D space and merged together into a triangular soup. Using a combination of both visibility and photo-consistency constraints, we filter the soup to further enforce its the consistency with the surface of the scene. Finally, a mesh is computed from the triangle soup using a reconstruction method that combines restricted Delaunay triangulation and Delaunay refinement. Visual and quantitative assessments applied on various large-scale datasets indicate the good performance of our algorithm compared to the state of the art methods in terms of accuracy of the mesh representation.

During the triangle filtering stage, it is sufficient for a triangle to not satisfy one of the constraints, to discard it from the triangle soup. As a result, the final triangle soup might contain some holes, thus, not cover the entire surface of the object leading to incomplete 3D meshes. This limitation is clearly visible in Figure 6.10.

To overcome this limitation, one solution would be to consider using a hole-filling heuristic on the resulting mesh. One common approach is to triangulate each connected component of the surfaces boundary, thus filling each hole with a patch that has the topology of a disc. We however would rather consider taking inspiration in the very recent work of [Toldo & Fusiello, 2010] and fill the gaps through, heuristically, at the triangle soup construction stage. Also, to get more accurate and visually pleasing results we want to consider taking inspiration in the work of [Vu et al., 2009] and possibly add a post-processing variational refinement step to our generated meshes. Finally, we would like to incorporate in the reconstruction process a scale dependent sharp edges detection and recovery procedure to build geometrically simplified models.

[3] http://certis.enpc.fr/index.html

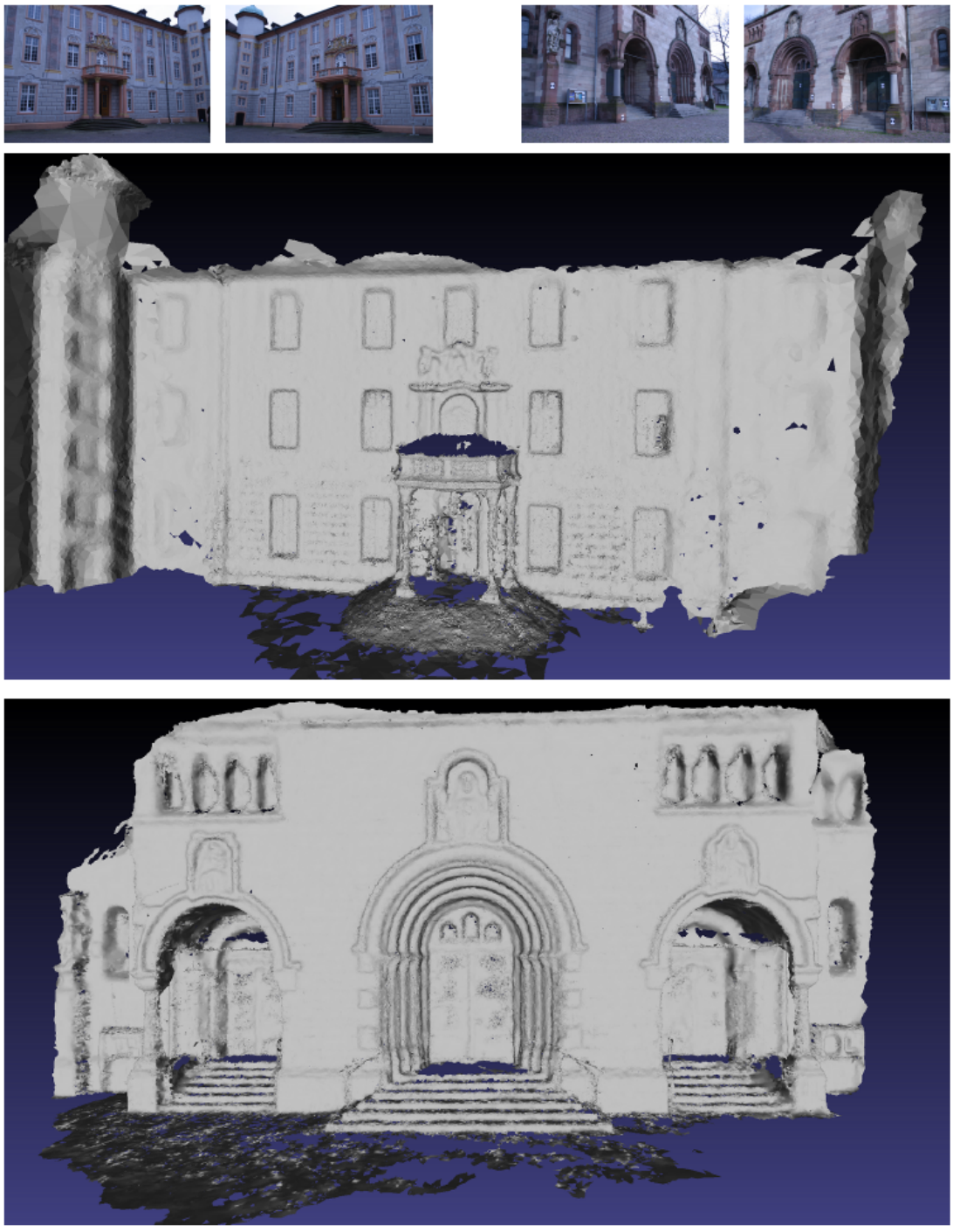


Figure 6.10: Top to bottom: two images of *entry-p10* and *Herz-Jesu-p25* datasets provided by [Strecha et al., 2008]; our reconstruction of *entry-p10* (500 000 triangles); our reconstruction of *Herz-Jesu-p25* (1 200 000 triangles).

scene	mesh		**times** (TS,R)	figure
	3D points	triangles		
entry-p10 *243 130 pts*	223 300	503 465	(5,4)	6.10
fountain-p11 *392 489 pts*	231 884	503 591	(8,4)	6.8
Herz-Jesu-p25 *712 105 pts*	501 788	1 200 755	(25,13)	6.10
Aiguille-du-Midi *6 788 290 pts*	131 202 465 419	272 792 992 140	(240,9) (240,20)	6.11

Table 6.1: Statistics of model reconstruction examples. Running times are in minutes, (TS,R) denote respectively the triangle soup extraction and mesh generation.

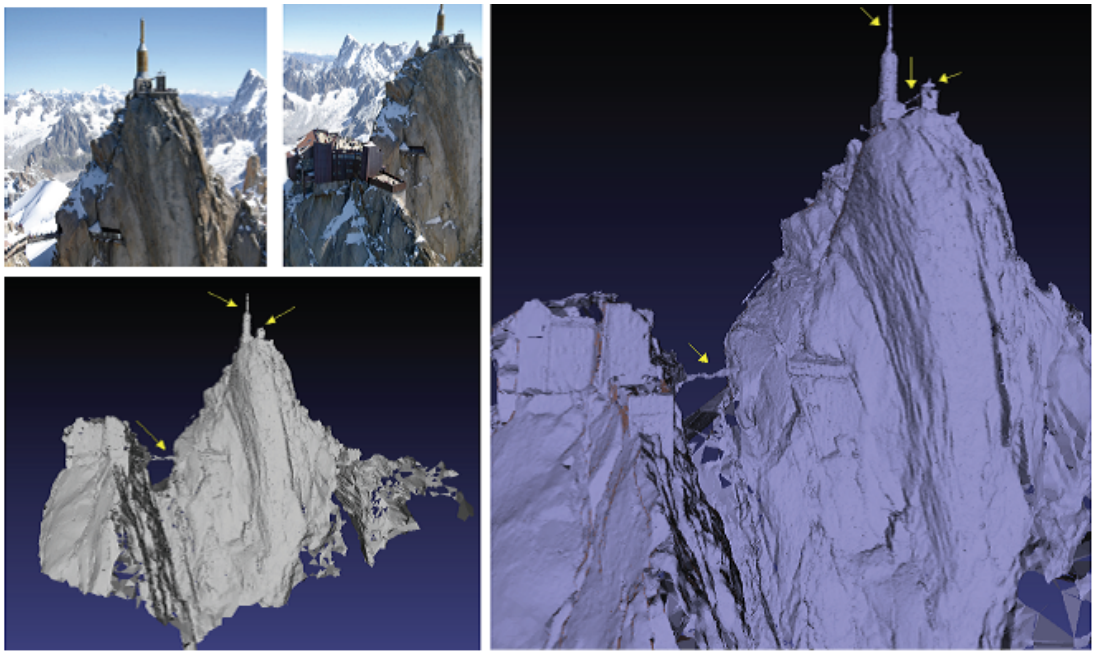


Figure 6.11: On the left: two images of the *Aiguille du midi dataset* (©B.Vallet/IMAGINE); below: our low resolution reconstruction (270 000 triangles). On the right: our high resolution reconstruction (1 000 000 triangles). Note how most details pointed by yellow arrows are recovered in both resolutions.

6.7 Publications

A preliminary version of this work appeared as a video in the ACM Symposium on Computational Geometry 2009 [Salman & Yvinec, 2009] and as a short paper in the Asian Conference of Computer Vision Workshop on Representation and Modeling of Large-Scale 3D Environments (Modeling-3D) 2009 [Salman & Yvinec, 2009] followed by an extended version in the International Journal of Virtual Reality [Salman & Yvinec, 2010].

Part of this work has also led to the development of the point set processing CGAL package [Alliez et al., 2010b].

The oldest, shortest words
– "yes" and "no" – are those
which require the most thought.
Pythagoras (582 - 497 BC)

7

Piecewise smooth surface reconstruction

We address the problem of generating quality surface triangle meshes from 3D point clouds sampled on piecewise smooth surfaces. Using a feature detection process based on the covariance matrices of Voronoi cells, we first extract from the point cloud a set of sharp features. Our algorithm also runs on the input point cloud a reconstruction process, such as Poisson reconstruction, providing an implicit surface. A feature preserving variant of a Delaunay refinement process is then used to generate a mesh approximating the implicit surface and containing a faithful representation of the extracted sharp edges. Such a mesh provides an enhanced trade-off between accuracy and mesh complexity. The whole process is robust to noise and made versatile through a small set of parameters which govern the mesh sizing, approximation error and shape of the elements. We demonstrate the effectiveness of our method on a variety of models including laser scanned datasets ranging from indoor to outdoor scenes.

7.1 Introduction

7.1.1 Motivation

Surface reconstruction consists in inferring the shape of an object from a 3D point cloud sampled on the surface. The problem of automatic surface reconstruction from point samples has received considerable amount of efforts in the past twenty years. Surface reconstruction problems occurs in different flavors according to the quality of the data which are more or less entangled with noise, and the properties of the reconstructed object whose surface can be smooth or structured with sharp features. Moreover, an implicit description of the reconstructed object is enough for some applications, whereas some other require a data structure more amenable to further processing, like a mesh.

In this chapter, we focus on the case where the sampled surface is piecewise smooth and the output sought after is a high quality triangle mesh approximating the surface. Our goal is to generate from the input point cloud a mesh including a faithful representation of sharp edges, thus, improving the trade-off between the accuracy and the size off the mesh. We propose for that an algorithm which experimentally proves to be fairly robust to noise.

7.1.2 Related work

Reconstruction and quality surface mesh generation are often tackled separately in the literature. We first review the approaches dealing with the problem of robust reconstruction of piecewise smooth surfaces, before reviewing the approaches that combine reconstruction and quality surface mesh generation.

Towards piecewise smooth reconstructions.

In the quest to achieve both noise robustness and recovery of sharp features, one can identify reconstruction approaches ranging from smooth to sharp through "nearly"-sharp.

Smooth surfaces. As noise and sparseness are common in laser scanner datasets, we have seen in recent years a variety of approaches for robust reconstruction of smooth surfaces. A popular approach consists of searching an implicit function whose zero level set fits the input data points. The searched implicit functions can take different forms, being for instance approximated signed distance functions to the inferred surface [Ohtake et al., 2005a, Guennebaud & Gross, 2007] or approximated indicator functions of the inferred solid [Kazhdan et al., 2006]. Implicit functions methods often provide spectacular robustness to noise. However most of them assume that the data points come with oriented normals and are sampled from a smooth surface. The latter assumption leads to erroneous surface reconstruction at locations where the smoothness assumption is not met.

> **Note:** A more detailed survey on smooth surface reconstruction methods is available in Chapter 4.

An intermediate step toward handling sharp features consists of using locally adapted anisotropic basis functions when computing the implicit function [Dinh et al., 2001]. In their approach, [Adamson & Alexa, 2006] perform piecewise polynomial approximation through anisotropic moving least-squares (MLS). More recently, [Oztireli et al., 2009] used a kernel regression technique to extend the MLS reconstruction to surfaces with sharp features. While satisfactory for some applications (e.g., visualization as opposed to mesh generation), these approaches generate surfaces which are more accurate around sharp features but are still smooth. In addition, boosting the anisotropy of a function is a dangerous game when simultaneously targeting robustness to noise, as such a strategy may amplify noise instead of the true features (features and noise are ambiguous as both are high frequency). Furthermore, a purely *local* distinction between smooth parts and sharp edges is likely to yield fragmented edges while a global feature extraction approach can favor long edges in order to recover the true structure.

Handling sharp features. This brings out another thread of work aimed at extracting long sharp features. When the input point cloud is endowed with normals, perceptual grouping rules can be used to infer smooth surface patches [Guy & Medioni, 1997] and sharp features may then be recovered as intersections of adjacent patches. When normals are not provided, one can use a multi-scale approach to decide if a point is on a feature [Gumhold et al., 2001, Pauly et al., 2003], and construct a minimum-spanning tree to recover the approximate feature graph. A couple of years later, [Demarsin et al., 2007] used PCA to estimate the normals of the input sample points, the latter are then clustered based on the normal variation in local neighborhood. Points near the cluster boundaries are tagged as feature points. Smooth approximations of the sharp features are then obtained by fitting smooth curves from the minimal spanning tree of the tagged data points. In their work, [Daniels et al., 2007] designed an approach to extract feature-curves using a method based on a robust MLS projection operator

which locally projects points onto the sharp edges, and grow a set of polylines through the projected points. An alternative approach was suggested by [Jenke et al., 2008] as they extract the feature lines by robustly fitting local surface patches and by computing the intersection of close patches with dissimilar normals. Both methods show satisfactory results even for defect-laden point sets. More recently, [Weber et al., 2010] proposed a fully automatic method that uses *Gauss map clustering* for feature detection. This method does not rely on local surface reconstructions, and no normal information is required.

However, it should be noted that none of the of the feature estimation methods cited above come with theoretical guarantees. Recently, [Merigot et al., 2009] proposed a robust estimation of curvature tensors based on a covariance measure defined from the data points Voronoi cells. Their method provides theoretical guarantees on robustness under bad sampling and noise conditions (this method will be described in more details in Section 7.2.1).

One way to make a local, binary distinction between smooth parts and sharp features consists of performing a local clustering of the inferred normals to the surface [Ohtake et al., 2003a]. If this process reveals more than one cluster, the algorithm does not fit just one low-degree implicit function, but as many quadrics as the number of clusters. This leads to faithful local reconstruction of a sharp edge as the intersection of two implicit primitives in the case of two clusters. A corner is reconstructed as the intersection of three or more primitives. Note in passing that the method misses tips (sharp point that are not incident to any sharp crease), cusps and darts.

In order to achieve improved robustness, [Fleishman et al., 2005] segment neighborhoods of points by growing regions belonging to the same part of the surface. Robustness is obtained by using a *forward search* technique which finds reference planes corresponding to each segmented region. While constituting an important progress, the method requires very dense point clouds, and is rather expensive. Also, potential instabilities in the classification can create fragmented surface parts. Targeting even higher robustness, [Lipman et al., 2007] enrich the MLS projection framework with sharp edges by defining an indicator field for local singularities based on the error of the MLS approximation. However, they choose to restrict their approach to a single singularity within each point neighborhood, which significantly limits the type of feature points that can be handled.

Some other approaches [Reuter et al., 2005, Guennebaud & Gross, 2007] require some explicit user interaction to enrich the input point cloud. Their projection operators take into account the sharp features building upon the user-defined tags.

Coupling reconstruction and mesh generation.

One popular approach that couples reconstruction and mesh generation is the extended marching cubes introduced by [Kobbelt et al., 2001]. The input is assumed to be a discretized signed distance function enriched with oriented normals, the latter being used to decide whether or not a voxel contains a sharp feature. If a voxel is classified as containing a feature, some additional vertices are constructed within the voxel and placed at intersections between the planes defined by the vertices and their affiliated normals. This approach works robustly and efficiently by truly coupling feature extraction and mesh generation. Nevertheless, it relies on the marching cubes process which is well known to produce rather poor quality surface meshes.

Two approaches which are able to generate quality surfaces while preserving the features [Daniels et al., 2007, Jenke et al., 2008] proceed sequentially by first extracting a set of features, then using the advancing front method proposed by [Schreiner et al., 2006]. The latter

produces quality isotropic meshes from different inputs including MLS representations, and can control the density of the mesh by pre-computing a guidance field. The algorithm uses time-consuming pre-processing of the point cloud in order to construct a guidance field, as well as a final post-processing using MLS projections.

A relevant alternative to the marching cubes and advancing front strategies is the Delaunay refinement paradigm [Boissonnat & Oudot, 2005]. In this method the surface mesh is intersection free by construction as filtered out of a 3D Delaunay triangulation. A number of additional guarantees are also provided after termination of the refinement process, such as a good shape of elements, a faithful approximation of geometry and normals, and a low complexity of the mesh. More interestingly in our context, Delaunay refinement is able to couple reconstruction and mesh generation. At the intuitive level, the refinement procedure is combined with a sensing algorithm probing an implicit surface defined from the data points. The probing is performed along Voronoi edges of sampling points which are not only longer - hence more effective at probing - than the short edges of the marching cubes but also data-dependent as they become more and more orthogonal to the sensed surface as the refinement process goes along.

However, in [Boissonnat & Oudot, 2005] sharp features are not explicitly handled, hence they are not accurately represented in the output mesh. One way to circumvent this drawback is to extract the feature lines and to constrain the Delaunay refinement to preserve them. However, it is well known that the presence of constrained edges where meeting surface patches form small angles, endangers the termination of the Delaunay refinement [Shewchuk, 2000]. Recently, [Cheng et al., 2007b, Cheng et al., 2007a] propose to deal with this problem using the method of protecting balls set up around sharp features. Protecting balls prevent the insertion of refinement no point is inserted into these protective balls.

Although not directly fit into the following classification, it is worth noting that a fully automatic feature-sensitive remeshing technique was designed recently by [Lévy & Liu, 2010] and compares favorably with Dey *et al.* Delaunay refinement and protecting balls approach. They use an anisotropy field oriented along the input mesh facets normals in a so called L_p *Centroidal Voronoi Tesselation* objective function to recover sharp features without any need to tag them manually. Their approach has the nice property of optimizing the mesh vertices locations resulting in nearly equilateral facets of homogeneous sizes.

Note: A more extensive survey on algorithms for mesh generation is available in Chapter 5.

7.1.3 Contribution

In what follows we argue that some modifications of the implicit smooth surface reconstruction algorithms can produce results with accuracy on par with the best current piecewise smooth reconstruction methods. Given a 3D point cloud sampled on a piecewise smooth surface, our algorithm consists of two steps. In the first step, we extract from the input 3D point cloud a set of polylines approximating the sharp edges of the object. This feature extraction is based on the robust feature estimation framework of [Merigot et al., 2009]. In parallel, our algorithm runs on the data points a reconstruction process, providing an implicit surface, *i.e.* an implicit function whose zero level sets approximates the data points. The second step of our algorithm generates a mesh approximating the implicit surface while including a faithful representation of the extracted sharp edges. This step runs a feature preserving version of the Delaunay refinement meshing algorithm, based on the meshing algorithm of [Boissonnat & Oudot, 2005] and using the mechanism of protecting balls introduced by [Cheng et al., 2007b, Cheng et al., 2007a]. An overview of the method is illustrated in Figure 7.1.

Figure 7.1: Overview of the method at a glance. From left to right: given a point cloud sampled on a piecewise smooth surface; detect the potential sharp edges points, cluster them with respect to the underlying sharp features direction, and extract explicit polylines for each cluster; sample protecting balls on the extracted polylines; combine extracted polylines to implicit surface reconstruction through feature preserving mesh generation to output a 3D mesh where the sharp edges are explicitly tesselated and appear as a subset of the triangle edges.

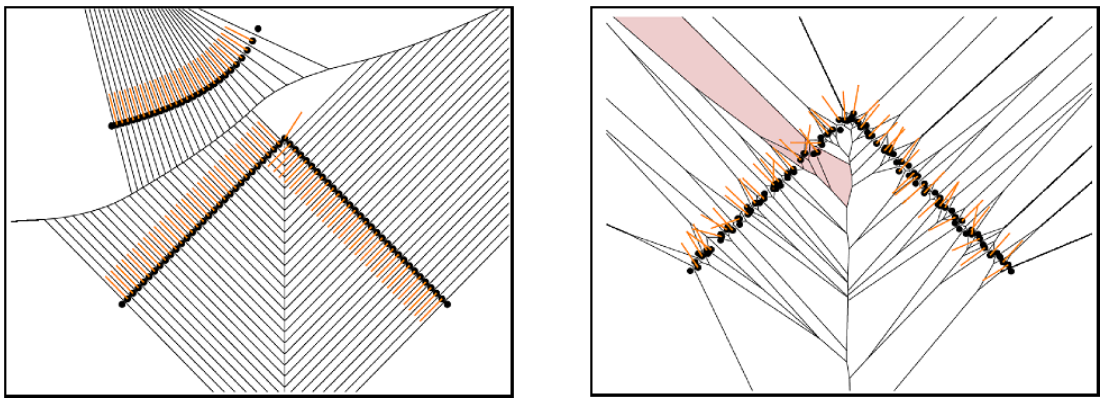


Figure 7.2: The shape of Voronoi cells depends on the whole set of data, and some cells are unbounded (left), and is very sensitive to noise (right) (*courtesy of Maks Ovsjanikov*).

The technical tools used in this algorithm are directly steered by recent work [Merigot et al., 2009, Cheng et al., 2007a]. However, they have not been used previously in this combination, and we argue that it is this particular synthesis of existing ideas which is the key of its success.

The main benefits of our algorithm are:

- **Accuracy.**
  The method provides a meshed representation of a point sampled object with an enhanced trade-off between accuracy and mesh size.
- **Robustness.**
  The method is robust to noise and stable under sparse sampling.
- **Flexibility.**
  The method offers considerable flexibility as it can be tuned with respect to the scale and the targeted size of the mesh.

The remainder of this chapter is structured as follows: We first describe our feature extraction process (Section 7.2) before we explain our feature preserving mesh generation (Section 7.3.1) in detail. We then describe our implementation and the datasets used for our tests and show results for a large number of parameter settings (Section 7.5) before we conclude. Please refer to Chapter 2 for a detailed description of the geometric concepts used in our method.

7.2 Feature extraction

7.2.1 Feature detection

A point on a piecewise-smooth surface, is called a "sharp" feature point, if the surface has no tangent plane at this point. The locus of feature points on a piecewise-smooth surface has the structure of a graph.

We are given a point cloud $\mathcal{P}$ sampling a piecewise-smooth surface $\mathcal{S}$. In order to allow the mesh generation algorithm to faithfully reproduce the sharp features of $\mathcal{S}$, we need to extract from the input point cloud an approximation of the feature graph of $\mathcal{S}$. Our first goal is to identify the data points that are close to sharp edges. To this aim, we use the method of

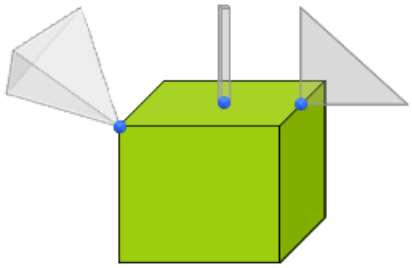


Figure 7.3: The Voronoi cell of a point on the surface of a cube is pencil, flat or cone shaped depending on the number of "small" eigenvalues of its covariance matrix.

[Merigot et al., 2009] based on an analysis of the shape of Voronoi cells through their covariance matrices.

As remarked in [Alliez et al., 2007], information about the normal directions of the underlying surface can be obtained through eigenanalysis of the covariance matrices of the Voronoi cells of the points that sample this surface. Recall that the covariance matrix of a bounded domain E of $\mathbb{R}^3$ with respect to a base point p is defined by

$$\mathrm{cov}(E,p) = \int_E (x-p)(x-p)^T \mathrm{d}x\,.$$

The eigenvectors of this matrix capture the principal axes of E while the ratio of the eigenvalues gives information on the anisotropy of E.

By construction, the shape of Voronoi cells depends on the whole set of data, and some cells are unbounded (see Figure 7.2 left). In order to get more local information, [Merigot et al., 2009] introduce an offset parameter R and consider for each data point p of $\mathcal{P}$, the covariance matrix, $M(p,R)$, of the intersection of the Voronoi cell $V(p)$ with a ball, $B(p,R)$, of radius R centered at p:

$$M(p,R) = \mathrm{cov}\left(V(p) \cap B(p,R),\ p\right)$$

The principal drawbacks of Voronoi-based methods is their high sensitivity to noise (see Figure 7.2 right). To alleviate the effect of noise, [Merigot et al., 2009] smoothes the information contained in the covariance matrices using convolution. The method computes for each point p the *convolved Voronoi covariance matrix*, CVCM for short, $M^c(p,R,r)$, by summing the covariance matrices of all points q of $\mathcal{P}$ at a distance less than r to x:

$$M^c(p,R,r) = \sum_{q \in B(p,r)} M(q,R)$$

The crucial remark is that, as it is illustrated in Figure 7.3, sample points lying on smooth surfaces, called *smooth points*, have a pencil shaped Voronoi cell; hence their CVCM has two "small" eigenvalues and a larger one, with the eigenvector corresponding to the large eigenvalue directed along the surface normal. Sample points that are close to sharp edges, called *edge points*, have rather flat Voronoi cells; therefore there CVCM has a single "small" eigenvalue and two larger ones, and the eigenvector corresponding to the small eigenvalue is directed along the edge. Sample points close to feature graph nodes, called *corner points*, tend to have cone-shaped Voronoi cells thus all three eigenvalues have comparable values.

Based on those observations, the algorithm we use to detect feature points (edge and corner points) is the following. We compute the CVCM, $M^c(p,R,r)$, of each point $p \in \mathcal{P}$, and sort the

Algorithm 1 Detect potential sharp edges points in $\mathcal{P}$

Input: a set of 3D points $\mathcal{P}$
Output: a set $\mathcal{F}$ of pairs of points $(p, e_2(p))$, $p \in \mathcal{P}$ and $e_2(p)$ is the edge direction.
for all points $p \in \mathcal{P}$ **do**
  Compute the Voronoi covariance measure M
for all points $p \in \mathcal{P}$ **do**
  Compute the convolved Voronoi covariance measure M^c and diagonalize it.
  Compute the ratio λ_0/λ_1
  if $\lambda_0/\lambda_1 < t_1$ **then** {p is on an edge or corner.}
    Compute the ratio λ_1/λ_2
    if $\lambda_1/\lambda_2 \geq t_2$ **then** {p is on an edge.}
      $\mathcal{F} \leftarrow (p, e_2(p))$

eigenvalues of this matrix by decreasing order $\lambda_0(p) \geq \lambda_1(p) \geq \lambda_2(p)$. If the ratio $\lambda_0(p)/\lambda_1(p)$, is greater than a threshold t_1, p is a smooth point. Else, if the ratio between $\lambda_1(p)/\lambda_2(p)$, is greater than a threshold t_2, p is an edge point. Else, point p is a corner point. We gather, in a set $\mathcal{F}$, the pairs $(p, e_2(p))$, where p is a point classified as an edge point and $e_2(p)$ is the eigenvector of its CVCM corresponding to the smallest eigenvalue $\lambda_2(p)$, which is supposed to be aligned with the sharp edge direction. Note that we choose not to keep in $\mathcal{F}$, the pairs corresponding to corner points because of the instabilities of the data around these positions. Instead, we will recover corner points as extremities of sharp edges, as we will see in Section 7.2.3.

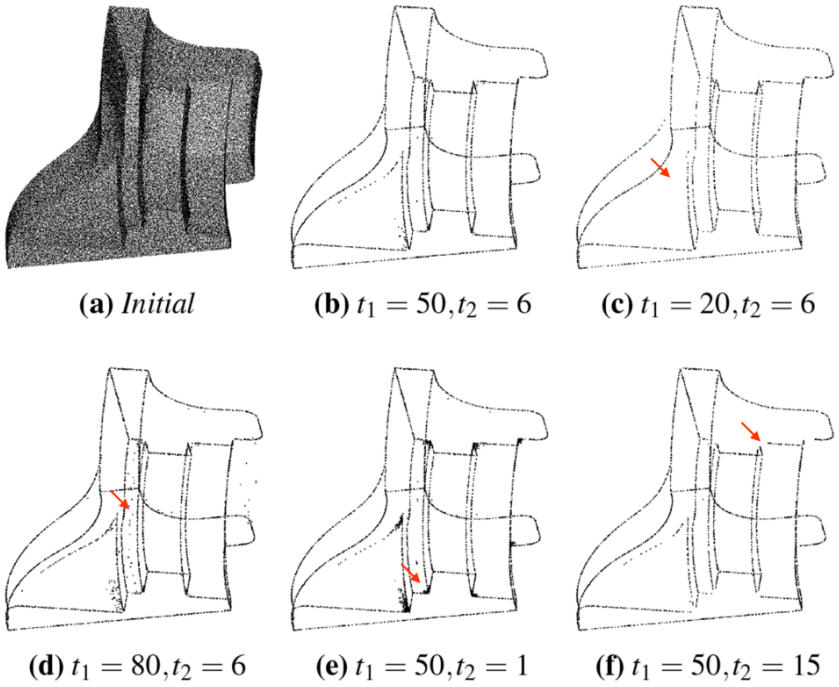


Figure 7.4: Detected edge points on the *fandisk* model (*courtesy of Joel Daniels II*), with various threshold t_1 and t_2 values. Note the differences with (b) marked by a small arrow.

Parameters - The method needs four parameters: the offset radius R, the convolution radius r and both thresholds t_1, t_2; and can be summarized by the pseudo-code in the Algorithm 1. Parameter R should be chosen smaller than the *external reach* of the underlying surface defined as the smallest distance from a point of the surface to the external medial axis. Parameter r has two different roles: besides smoothing the information, it determines the curvature radius under which a curved area is considered as a sharp feature. Consequently, it should be chosen depending on the noise level and on the sizing field of the targeted mesh. In all our experiments, we fixed parameter R to be one tenth of the point cloud bounding sphere radius and r to be a tenth of R, which leads to good results. Moreover, we allow the user to modify t_1 and t_2 manually, using an interactive interface. Figure 7.4, shows an example of the effect of the choice of parameters t_1 and t_2 on the set of detected edge points.

7.2.2 Feature clustering

In [Merigot et al., 2009], it is proven that the convolved covariance matrix is robust to noise, by deriving a bound on the quality of the results as a function of the Hausdorff distance between the point cloud and the sampled surface. However, the reconstruction of the sharp feature graph is left unaddressed in this work. Here, we need to approximate the sharp edges with polygonal lines. In their work [Daniels et al., 2007] presented a technique for solving a similar problem. We however proceed differently taking advantage of the edge direction information we get from the CVCM analysis.

Our goal here is to group edge points of $\mathcal{F}$ into clusters, where each cluster samples a straight or bending sharp edge. For that, we need a similarity measure that will quantify if two edge points belong to the same sharp edge.

A pair (p_i, p_j) of edge points is said to be a close pair if the Euclidean distance $d(p_i, p_j)$ is less than $\rho_k(p_i)$, where $\rho_k(p_i) = \frac{1}{k}\sqrt{\sum_{l=1}^{k} |p_i - p_l|^2}$, the sum running on the k nearest neighbors to p_i in $\mathcal{F}$. Two close points $p_i, p_j \in \mathcal{F}$ belong to the same cluster if the angles $\alpha_i = \angle(e_2(p_i), p_i p_j)$ and $\alpha_j = \angle(e_2(p_j), p_i p_j)$ are smaller than a given threshold θ.

This criteria is injected in an union-find algorithm (Algorithm 2) that partitions points of $\mathcal{F}$ into clusters $\mathcal{F}_i$. In addition, if only one of the angular conditions is satisfied for a close pair (p_i, p_j), say $\alpha_i < \theta$ but $\alpha_j \geq \theta$, point p_j is classified as an *extremity point* of cluster $\mathcal{F}_i$ and included in a set $\mathcal{E}_i$ of extremity points related to $\mathcal{F}_i$. These extremity points will be of use to elongate the constructed polylines in Section 7.2.3, facilitating the resolution of the polylines junction recovery problem. Moreover, our implementation prunes clusters consisting of few points ($|\mathcal{F}_i| \leq 6$) and unlabels the potential extremity points associated to them.

Parameters - The identification of good clusters relies on an appropriate selection of parameters θ and k. For big values of θ the detected clusters will be thick, thus it will be hard to sort the clustered points, a step we need for recovering the features. For our purposes, $\theta = 15°$ and $k = 20$ provided sufficiently thin and representative clusters to facilitate the feature recovery procedure (Figure 7.5), without the need to apply the thinning heuristics of [Lee, 2000].

Algorithm 2 Clustering points in $\mathcal{F}$

Place each point $p_i \in \mathcal{F}$ in its own cluster $\mathcal{F}_i$ with it's potential extremity points $\mathcal{E}_i = \emptyset$
for all points p_i **do**
  for all $p_j \in \mathcal{F}$ such that $d(p_i, p_j) < \rho_k(p_i)$ **do**
    if p_j and p_i are NOT in the same cluster ($\mathcal{F}_j \neq \mathcal{F}_i$) **then**
      Compute angle $\alpha_i = \angle(e_2(p_i), p_i p_j)$
      Compute angle $\alpha_j = \angle(e_2(p_j), p_i p_j)$
      if $\alpha_i \leq \theta$ and $\alpha_j \leq \theta$ **then**
        Merge the clusters $\mathcal{F}_i$ and $\mathcal{F}_j$
        Merge $\mathcal{E}_i$ and $\mathcal{E}_j$
      if $\alpha_i \leq \theta$ and $\alpha_j > \theta$ **then**
        $\mathcal{E}_i \leftarrow p_j$

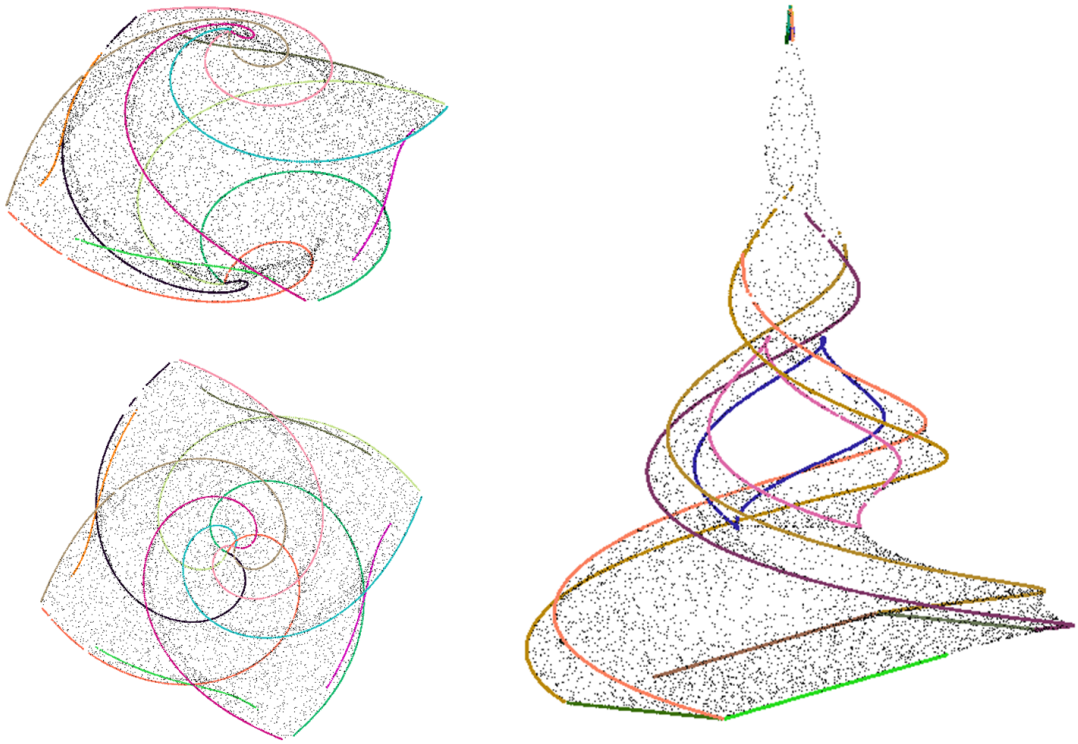


Figure 7.5: Result of the clustering step on the *octa-flower* (left) and *twirl* (right) models.

7.2.3 Feature recovery

In the following our aim is to recover from each cluster, provided by the previous step, a polyline that will best fit the shape of the underlying sharp edge.

For each cluster $\mathcal{F}_i$ we build a polyline $Poly(\mathcal{F}_i)$ by applying iteratively a procedure $succ(p)$ that discovers the successor of a point p (Figure 7.6 top). We start at a random seed point $p \in \mathcal{F}_i$. The successor $s(p)$ of p is defined as follows: we consider the subset $N(p) \subset \mathcal{F}_i$ of points included in a ball of radius r_p centered at p. All points in $N(p)$ are then projected onto the line defined by p and the edge direction $e_2(p)$. Points in $N(p)$ can be divided into two subsets by

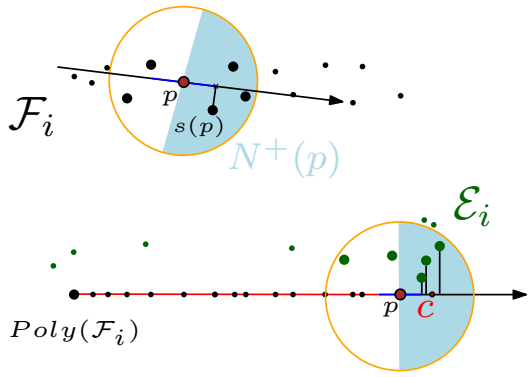


Figure 7.6: Illustration of feature recovery step. Top: procedure $succ(p)$ that discovers the successor, $s(p)$, of a point $p \in \mathcal{F}_i$. Bottom: after smoothing the polyline $Poly(\mathcal{F}_i)$, we elongate it at each endpoint p using the detected extremity points $\mathcal{E}_i$.

inspecting the position of their projections relative to p. If p is the starting point, we choose randomly one of the subsets to be $N^+(p)$. Otherwise, let $N^+(p)$ be the subset of points whose projections are in the direction opposite to the predecessor of p. Then, $s(p)$ is defined as the point in $N^+(p)$ with closest projection to p. The procedure $succ(p)$ is applied iteratively until an endpoint is reached, *i.e.* a point for which $N^+(p)$ is empty. The procedure is then restarted from the seed point in the opposite direction.

The polyline $Poly(\mathcal{F}_i)$ and the edge directions attached to its vertices are then smoothed using a standard PCA based smoothing process.

To *lengthen* the recovered feature polylines $Poly(\mathcal{F}_i)$, we use the detected potential extremity points $\mathcal{E}_i$ attached to each cluster. For each endpoint $p \in \mathcal{F}_i$, if $\mathcal{E}_i$ is not empty, we search for neighboring points $p_j \in \mathcal{E}_i$ such as the Euclidean distance $d(p, p_j) < r_p$. All such points are projected onto the line defined by p and the edge direction $e_2(p)$. Then, we compute the centroid c of the projected points located on the half-line defined by p and not containing the projection of $N(p)$. Recall that because p is an endpoint $N^+(p)$ is empty and $N(p) = N^-(p)$. Finally, point c is added to the polyline as the successor of p and attached a direction $e_2(c) = e_2(p)$. Point c is considered as the new endpoint of $Poly(\mathcal{F}_i)$ (see Figure 7.6 bottom).

Our implementation maintains for each polyline $Poly(\mathcal{F}_i)$ a value corresponding to the polyline length that will be of use in the feature junction recovery step to prevent joining the endpoints of a single short polyline.

Parameters - The recovered polyline result is dependent on the size of the selected neighborhood. The radius parameter r_p should be related to the *local feature size* of the polyline to be recovered. Optimally, one should study the effect of the neighborhood size and propose an optimal neighborhood-size-selection scheme. Practically, we do not have this information, hence we choose $r_p = \rho_k(p)$ where $k = 5$.

Junction recovery – In order to build a realistic approximation of the sharp feature graph, we need to construct appropriate junctions between the constructed polylines. The task is simplified as we have extended the polylines and thus reduced the search space for potential junctions. Note that sharp edges joined at vertices of the feature graph are classified as follows: a *dart* is incident to a single sharp edge; a *cusp* is incident to two sharp edges forming a sharp angle at the junction; a *corner* is incident to three or more sharp edges.

A natural solution is to merge in a single vertex, polylines endpoints that are spatially close.

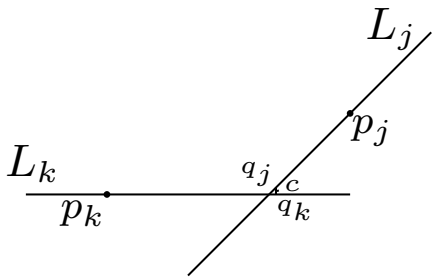


Figure 7.7: Sketch of the junction recovery step.

We could simply take as position for the merged vertex the barycenter of the merged endpoints. However, to increase the accuracy of the approximation of sharp feature junctions we proceed as follows:

For each endpoint p of $Poly(\mathcal{F}_i)$, we start by constructing the set $\mathcal{J}(p)$ of endpoints that are within a ball $B(p, r_{junc})$ of radius r_{junc} centered at p. In case there is no endpoint in $\mathcal{J}(p)$ other than p itself, we tag p as a dart and continue. Otherwise, $|\mathcal{J}(p)| \geq 2$ and we compute the junction position as follows. For every pair of endpoints $p_j, p_k \in \mathcal{J}(p)$, we consider the two points q_j, q_k which are the closest points such that q_j belongs to the line L_j passing through p_j and directed along $e_2(p_j)$ and q_k belongs to the line L_k passing through p_k and directed along $e_2(p_k)$. The junction point is computed as the barycenter of the set of closest points, except that each closest point that is not in the ball $B(p, r_{junc})$ is replaced by the endpoint p in the barycenter computation (See Figure 7.7).

Our procedure has the advantage of being easy to implement and is computationally inexpensive since it does not introduce any plane nor surface fitting and intersection computations. An example of finding junctions from polylines including three junction cases, darts, cusps and corners is illustrated in Figure 7.8. Note that we do not capture *tips*, *i.e.* vertices incident to zero sharp edges but are sharp.

Parameters - The radius parameter r_{junc} should be chosen large enough, in order to cover the empty space between the polylines endpoints. On the other hand, generally, r_{junc} shouldn't be larger than the length of the shortest polyline – which would lead to both of its endpoints being merged into a single junction. In practice, we found that choosing r_{junc} to be equal to five times the average spacing of the ending four points of a polyline leads to good results.

7.3 Feature preserving mesh generation

Besides extracting sharp edges, our algorithm runs on the data points a reconstruction process, such as Poisson reconstruction [Kazhdan et al., 2006] or any variant of moving least square [Guennebaud & Gross, 2007], providing an implicit description of a surface approximating the data points. Such an implicit surface is represented as a function ($\mathbb{R}^3 \to \mathbb{R}$) whose zero level set approximates the data points. Our goal in this section is to show how, using a feature preserving variant of a Delaunay refinement mesh generation algorithm, we obtain a surface mesh approximating the implicit surface and including a faithful representation of the sharp edges extracted in the previous Section 7.2.3.

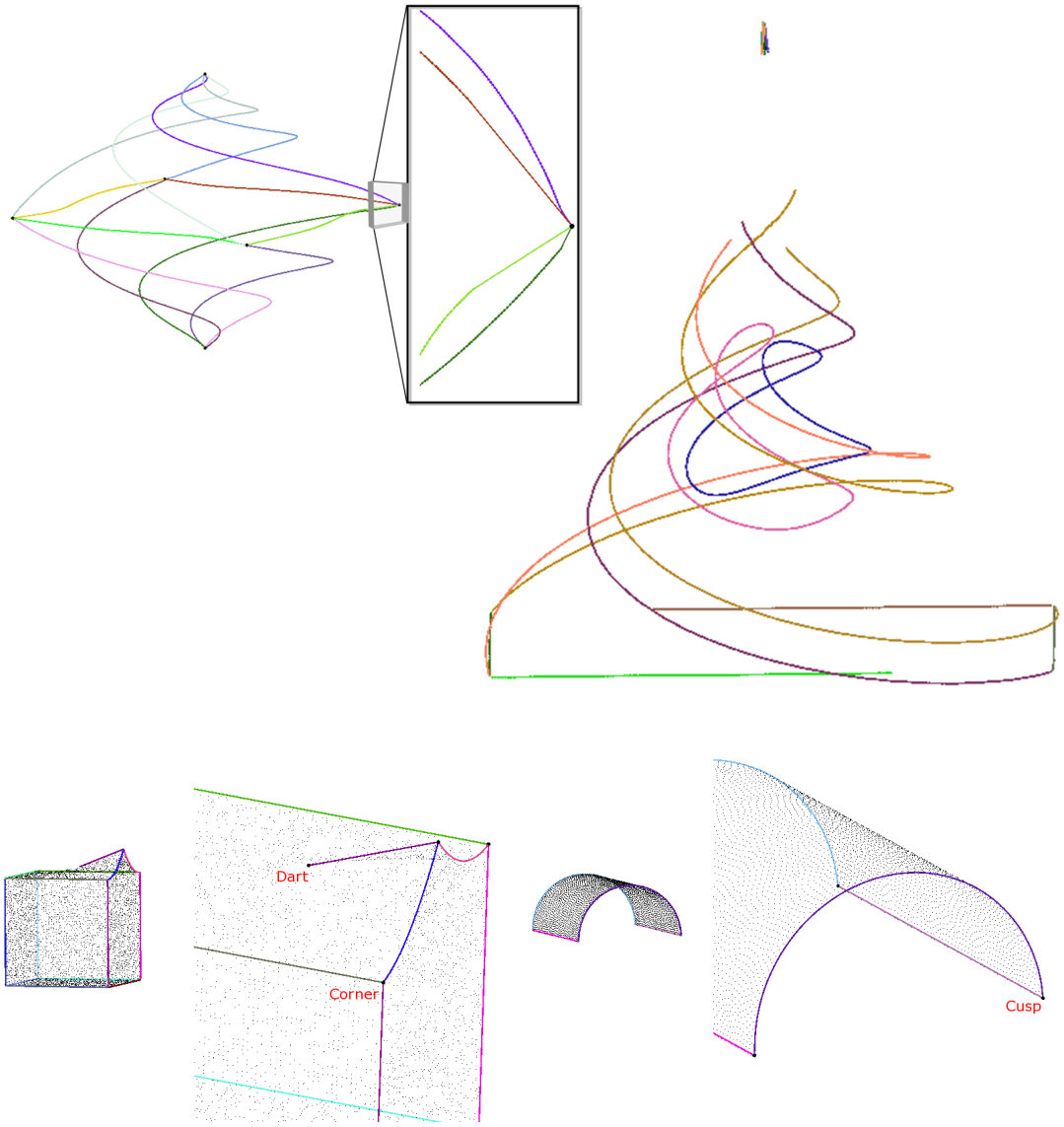


Figure 7.8: From Left to right, top to bottom: feature recovery for the *octa-flower* model and a closeup view of the recovered sharp edges and the recovered junction (black dot); recovered features for the *twirl* model; a dart and multiple corners are recovered from the *smooth feature* model; and multiple border cusps are recovered from the *half cylinder* model.

7.3.1 Delaunay refinement surface mesh generation

To generate a mesh approximating the implicit surface, we use the surface meshing algorithm of [Boissonnat & Oudot, 2005]. This algorithm is a Delaunay refinement, based on the notion of restricted Delaunay triangulation and described in more details in Section 6.2.

Such a Delaunay refinement surface meshing algorithm yields a quality surface mesh, free of self-intersections. However, since features are not handled explicitly, they are poorly represented in the output mesh (see Figure 7.12 middle).

Parameters - The parameters of the Delaunay refinement mesh generation algorithm are the parameters of the criteria defining the bad facets. Recall that a facet is bad if either some facet angle is smaller than α, or some facet edge is longer than l or the distance between the center of the facet and the surface is more than d.

7.3.2 Feature preserving extension

As already mentioned, the standard Delaunay refinement algorithm does not reconstruct sharp edges accurately. Moreover, enforcing these sharp edges to be in the output mesh, may affect the termination of the algorithm (Section 5.4). One clever way to overcome this limitation has recently been proposed by [Cheng et al., 2007b] and experimented in [Cheng et al., 2007a]. Their idea is to place *protecting balls* around the sharp edges and junctions, and then turn them to weighted points for further meshing. Building upon this technique, we use this mechanism to protect our detected polylines with balls before starting the Delaunay mesh refinement. Following [Cheng et al., 2007b], the protective balls have to satisfy the following properties:

- protecting balls are centered on sharp features and each feature is completely covered by the union of protecting balls centered on it;
- any two balls have an empty intersection, except balls with consecutive centers on a given sharp feature that intersect significantly but do not include each other's center;
- any three balls have an empty intersection.

Note: A similar feature preserving mesh generation strategy was employed by [Boltcheva et al., 2009] to generate high quality meshes from 3D multi-tissue medical images.

Once the protecting balls have been computed, they are regarded as weighted points and inserted altogether, as initial vertices, in a weighted Delaunay triangulation, *i.e.* regular triangulation. The Delaunay refinement process is then performed using this weighted Delaunay triangulation: at each step, the refinement process computes the weighted version of a bad facet surface center. This point is affected a zero weight and inserted in the weighted Delaunay triangulation.

Such a weight setting, together with the protecting ball properties, enforces the fact that two protecting ball centers, consecutive on a sharp feature polyline are guaranteed to remain connected by a restricted Delaunay edge [Cheng et al., 2007b]. Furthermore, owing to the fact that no three balls intersect, the refinement process never inserts refinement points in the union of the protecting balls nor probes the surface in this protected region. This ensures the termination of the refinement process whatever may be the dihedral angle formed by smooth patches incident on a sharp feature.

Protecting balls have a crucial influence on the sizing of the output mesh. On one hand, protecting ball centers define the approximation of sharp features in the final mesh, and the sizing parameter l of the Delaunay refinement cannot be smaller than the spacing between

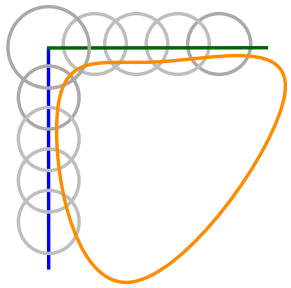


Figure 7.9: A 2D example of how protecting balls are placed. The union of protecting balls covers the gap between the implicit surface (orange) and the extracted feature polylines (blue,green).

protecting ball centers. On the other hand, as it is illustrated in see Figure 7.9, to provide a surface mesh with facets incident to the mesh edges approximating a sharp feature, the refinement process requires that the union of protecting balls covers the gap between the implicit surface and the extracted feature polylines.

In practice, to compute the protecting balls, we first keep all endpoint vertices of the detected polylines $Poly(\mathcal{F}_i)$ and then sample points on $Poly(\mathcal{F}_i)$ according to some user defined uniform distance d'. Then, the algorithm tries setting a uniform weight $w_p = 2/3 * d'$ on sample points. If the resulting set of balls do not comply with the above rules, it is locally fixed by adding new sample points and reducing ball radii.

Parameters - In our experiments, the mesh generation step depends on a single user-defined parameter δ. Indeed, we use as unit length the quantity avg defined as the mean of the distances from each input data point to each one of its six first nearest neighbors. Then, we fix the protecting ball parameter and Delaunay refinement parameters to: $d' = \delta * avg$, $\alpha = 25°, l = 2 * d', d = 0.6 * d'$. An example illustrating the effect of parameter δ on the output mesh resolution can be seen in Figure 7.10.

7.4 Implementation details

Our prototype is implemented in **C++** using, as in the previous chapter, the robust primitives provided by the CGAL library [cga, 2010]. We use CGAL 3D triangulation as the core data structure to compute the Voronoi covariance matrices of the input point cloud [Merigot et al., 2009]. The CGAL library also provides us with an implementation of the *Poisson reconstruction* method [Kazhdan et al., 2006] to compute an implicit surface from the input point cloud. Lastly CGAL provides an implementation of the Delaunay refinement surface meshing algorithm of [Boissonnat & Oudot, 2005].

In their work, [Merigot et al., 2009] described two practical approaches to compute the Voronoi covariance matrix at a scale R of a point cloud. Recall that the Voronoi covariance matrix at a scale R of a point cloud $\mathcal{P}$ is the measure $M(p, R)$, of the intersection of the Voronoi cell $V(p)$, where $p \in \mathcal{P}$, with a ball, $B(p, R)$, of radius R centered at p. Their first method to compute the VCM of a point cloud at a scale R, is a *Monte-Carlo approximation* that has the notable advantage of being easy to implement but is in practice slow at execution. The second algorithm they designed and use for their experiments is an *approximation by 3D tesselation*. It is much faster but is complex to implement.

(a) $\delta = 3.5$ (b) $\delta = 7$

Figure 7.10: Feature preserving mesh generation for the *fandisk* model. The threshold δ is used to adjust the size of the protective balls thus the mesh resolution; for (a) and (b), the meshed Poisson implicit function (PIF) is on the left whereas the meshes generated with our feature preserving extension using PIF are on the right. For both threshold values the extracted polylines and their endpoints are tesselated explicitly.

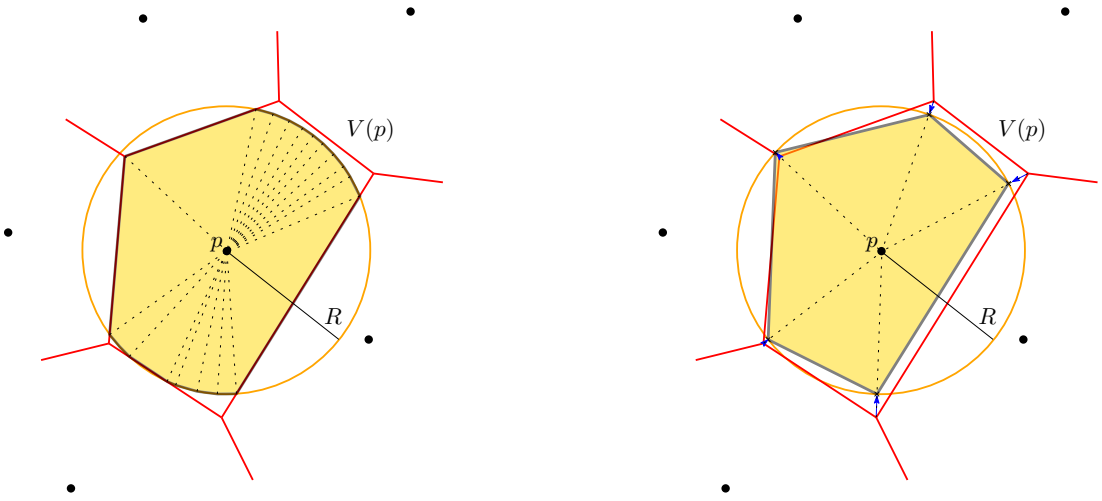


Figure 7.11: Two different methods for computing the Voronoi covariance matrix at a scale R. The intersection of the the Voronoi cell $V(p)$ with a ball $B(p, R)$ is approximated using: approximate tesselation of [Merigot et al., 2009] (left); our clamped approximation (right). Note how we project the vertices of $V(p)$ on the ball (blue arrows).

In our work we implemented an algorithm we call a *clamped approximation* that has the advantage of being faster than both aforementioned algorithms, while remaining very easy to implement. The principal idea to keep in mind is that covariance matrices have the nice property of being additive. Thus, one can simply decompose the intersection of the Voronoi cell with the ball, into tetrahedra and compute the covariance matrix of each tetrahedron analytically [Alliez et al., 2007]. For that, we project or "clamp" the vertices of each Voronoi cell onto the sphere of radius R centered at the same point. And then build tetrahedra by connecting each of the triplet clamped vertices to the center of the Voronoi cell. This can be done because the intersection of the Voronoi cell and the ball is star-shaped with respect to the center of the Voronoi cell. A sketch of our algorithm is illustrated in Figure 7.11 as well as the approximate 3D tesselation of [Merigot et al., 2009].

The output of this algorithm yields an approximate tetrahedralization of the intersection of the Voronoi cell with the ball of given radius R. We then use the closed-form formulas of [Alliez et al., 2007] to compute the covariance matrix of this approximate intersection.

A crucial component for reaching good timings is the efficient answer to queries for the subset of a point cloud contained in a given ball. Such a query is required for computing the CVCM used to detect edge points, it is also used for clustering edge points, and finally for recovering and joining the feature polylines. We have chosen to implement these queries using the ANN library [Mount & Arya, 2010]. Note that Poisson reconstruction assumes that the input points come with oriented normals. Here, we take advantage of the result from the eigenvector analysis of the CVCM. Each data point is attached a normal direction corresponding to the eigenvector of its CVCM with largest eigenvalue. The normals are oriented using the method described by Hoppe et al. in [Hoppe et al., 1992] and provided by the point set processing package [Alliez et al., 2010b] described in more details in Appendix A.

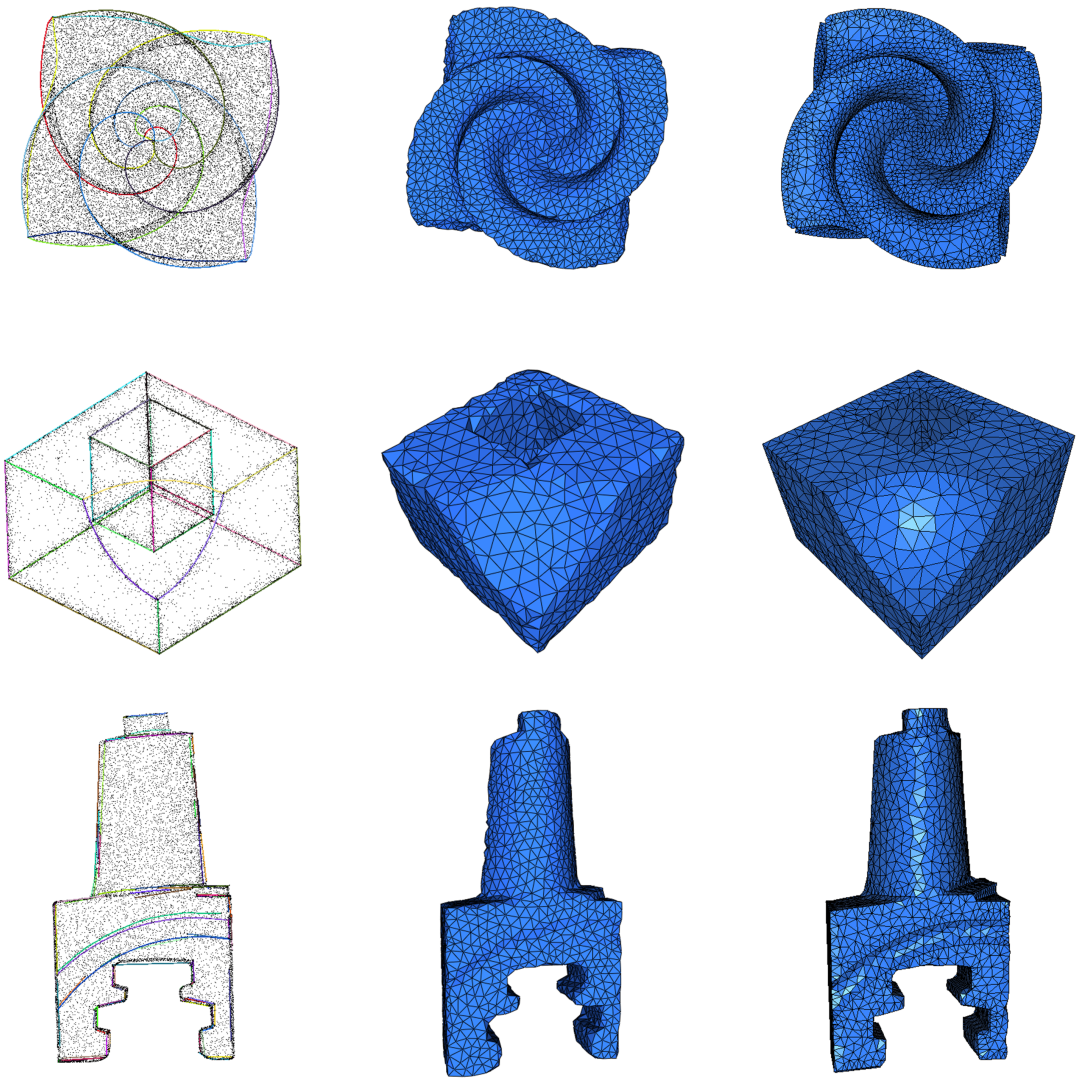
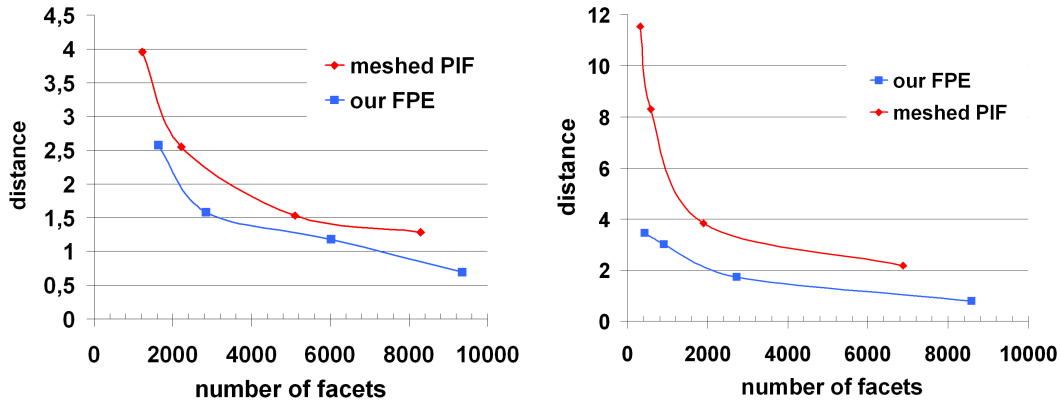


Figure 7.12: Feature preserving mesh generation on synthetic data sets. From left to right: Our extracted polylines overlayed to the input point cloud; Delaunay refinement mesh of the Poisson implicit surface; output of our feature preserving mesh generation. Top: *octa-flower*; middle: *carved_object* (*courtesy of Philipp Jenke*); and bottom: *blade* models.

7.5 Experimental results

Experiments where conducted on various datasets, we report here some of them: five "clean" synthetic models (*fandisk, octa-flower, carved object, blade, block*), a synthetic model with added noise (*fandisk*), the raw output of a triangulation-based Minolta^{TM}Vivid laser scanner for an indoor model (*Ramses*), and the raw output of a time-of-flight Leica^{TM}Cyrax laser scanner for an outdoor urban scene (*Church of Lans le Villard*). We refer the reader to Section 3.1 for more details on laser range scanning acquisition methods.

All results presented here have been produced using fixed values of the parameters that are



(a) *fandisk* (b) *block*

Figure 7.13: Quantitative evaluation of the accuracy versus mesh size :(a) the *fandisk* model, points on the curve correspond to values 10, 7, 4.5 and 3.5 of δ (b) the *block* model, points on the curve correspond to values 9, 6, 3 and 1.5 of δ. The Hausdorff distance plotted vertically is given in units of $\frac{1}{200}$ of the bounding box diagonal length.

mentioned in the **Parameters** paragraph at each of the algorithm steps, except for the three parameters t_1, t_2 and δ.

Parameters effect – In Figure 7.4 we illustrate the effect of tuning the parameters t_1 and t_2 for the feature detection step, on the *fandisk* model. Notice how t_1 acts on isolating smooth points from edge and corner points, whereas t_2 stirs the separation between edge and corner points.

Accuracy *versus* mesh size – Figure 7.10, illustrates the effect of parameter δ on the output mesh resolution. Here we have chosen $t_1 = 50$, $t_2 = 6$ and protected the extracted polylines with balls obtained for respectively $\delta = 3.5$ (a), $\delta = 7$ for (b).

We further evaluated our method on the three synthetic datasets, shown in Figure 7.12. The *octa-flower* model is interesting as it has curvy, sharp features. The two other models *carved_object*, and *blade* show the ability of our approach to recover sharp edges from sparse samplings ($\leq$ 50k points). Notice, that in the case of *carved_object* and *blade*, the original models are 3D meshes, but the vertices of those meshes are NOT lying on the feature lines. The initial triangle edges are zigzaggy along the feature lines. Again all comparisons are run with the same sizing field parameter δ, hence the same parameters for the Delaunay refinement algorithm.

As already mentioned, our goal is to improve the accuracy for a given mesh size, by ensuring a faithful representation of sharp edges. To illustrate our achievement with respect to this goal, we show, for each model in Figure 7.12, two meshes computed with the same value of parameter δ, hence the same value of the Delaunay refinement algorithm parameters ($\alpha = 25°, l = 2 * d', d = 0.6 * d'$).

For quantitative evaluation of the accuracy improvement, Figure 7.13 shows an approximation of the Hausdorff distance between the generated meshes and the ground truth model for various size of the generated mesh. Note that, the feature preserving Delaunay refinement (FPE) always outperforms standard Delaunay refinement (PIF) and that the effect is all the more important for coarse meshes.

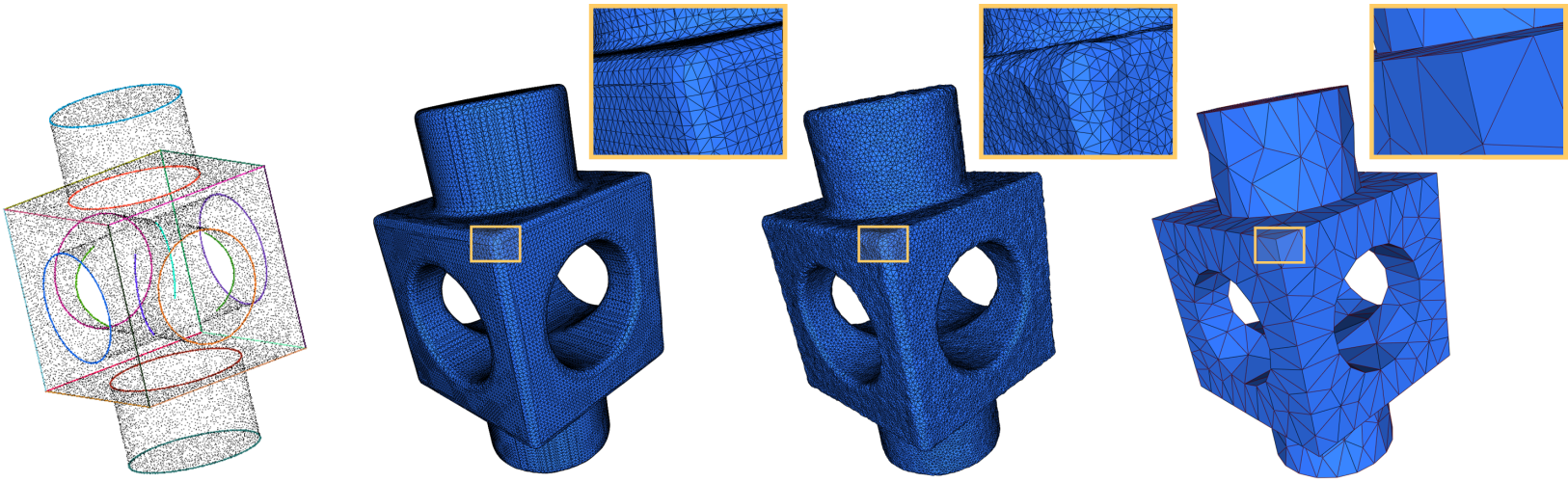


Figure 7.14: From left to right: input point cloud and overlayed extracted feature polylines for the *block* model and three generated meshes from the implicit surface obtained by Poisson reconstruction: marching cubes implementation [Kazhdan et al., 2006], with default octree depth, (62k facets); standard Delaunay refinement proposed by [Boissonnat & Oudot, 2005], with $\delta = 0.5$, (60k facets); our feature preserving Delaunay refinement, with $\delta = 6$, (900 facets).

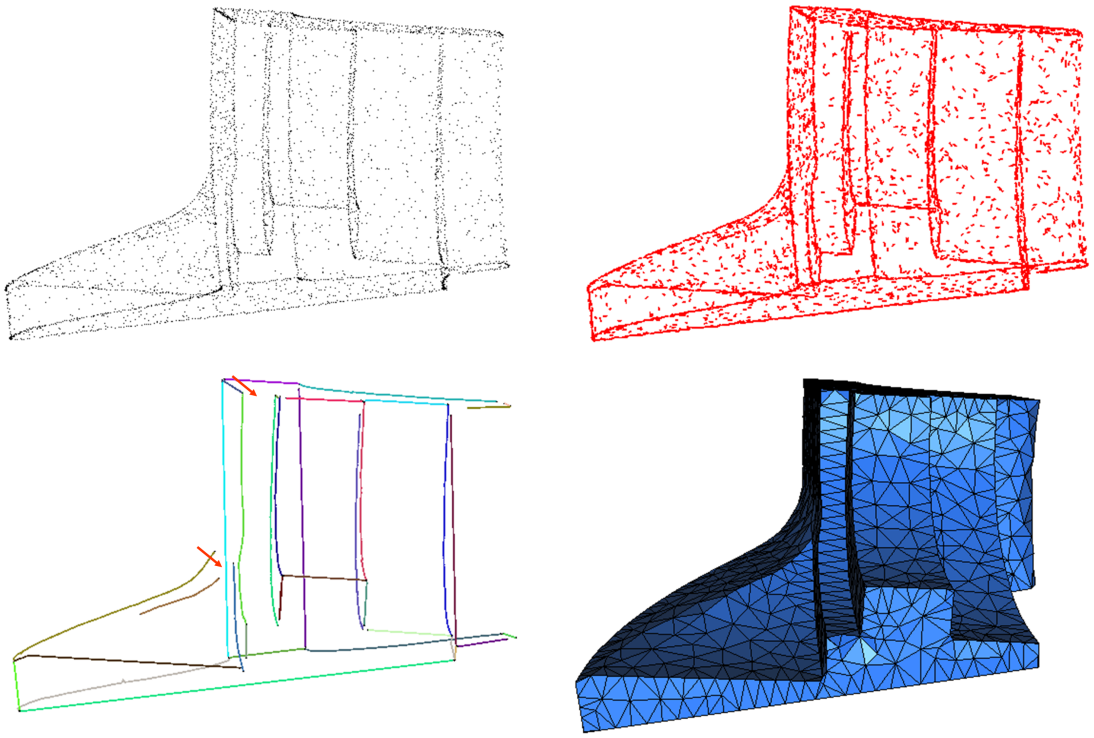


Figure 7.15: Feature extraction and preserving mesh generation for the *fandisk* model corrupted with uniform noise of 1% of its bounding box. From top to bottom, left to right: the detected edge points ($t_1 = 50, t_2 = 6$); the associated feature directions; the extracted feature polylines; the output mesh with ($\delta = 7$). Note how choosing large protective balls *patches* the small gaps (pointed by small arrows) between feature polylines.

Figure 7.14 is another illustration of the ability of our method to enhance the trade-off between accuracy and mesh size through feature extraction and preservation. Note that the only way for standard mesh generation methods (marching cubes and Delaunay refinement) to diminish the aliasing effect produced by the lack of feature lines is to generate high resolution meshes. However, as observed by [Kobbelt et al., 2001] oversampling will not solve the aliasing problem, since the surface normals in the reconstructed mesh will not converge to the normal field of the object.

Complex datasets – In order to evaluate the ability of our feature extraction method to cope with noise, we perturbed the points sampled on the *fandisk* model. The perturbation is uniform and has an amplitude of 1% of the diagonal of the dataset bounding box. Keeping the same set of parameters (t_1, t_2) as for the unperturbed *fandisk* model, Figure 7.15 shows the detected edge points. Although the edge points are more diffused with this amount of noise, the computed feature directions remain quite stable. Moreover, despite introducing strong noise, the feature extraction step constructed almost all sharp edges.

Note: Choosing large protecting balls, *i.e.* $\delta = 7$, forces some missing junctions between polylines to appear in the output mesh.

To evaluate the practical usability of this new approach, we applied it to two laser scan datasets acquired from two different environments. The first one is an indoor laser acquisition of the

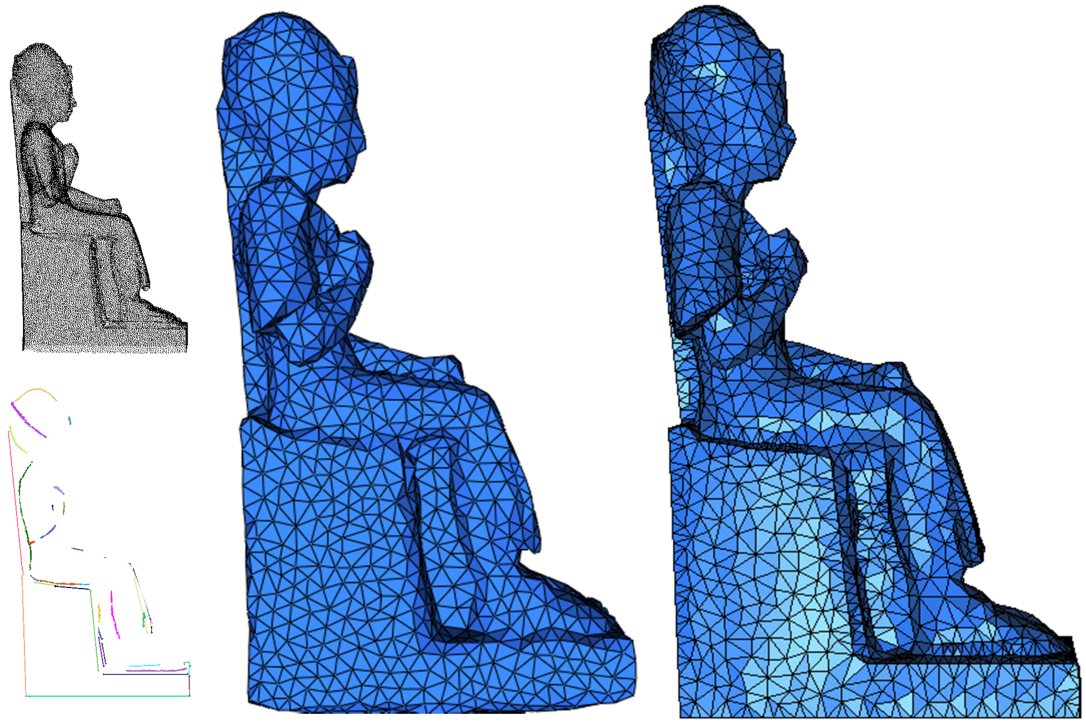


Figure 7.16: Left: input point cloud of *Ramses* model down-sampled to about 27% of the original dataset (top), extracted feature polylines (bottom). Middle: meshed Poisson implicit function. Right: output of our feature preserving mesh generation (5k facets).

Ramses model, Figure 7.16. The second is an outdoor laser acquisition of the *Church of Lans le Villard* in France, Figure 7.17. These datasets pose some problems compared to the synthetic models presented previously as near sharp features data are sparse and entangled with a high level of noise.

Note: Note that although Poisson reconstruction is not the most adapted surface reconstruction method for noisy datasets, our feature preserving output remains considerably more accurate and complete that the Delaunay refinement mesh of the Poisson implicit surface. We are confident that using other more robust to noise implicit surface reconstruction method [Mullen et al., 2010] will even provide better meshes.

The total time required to construct the feature polylines depends on the number of detected sharp edges points. Our prototype program has not been optimized and the details shown in Table 7.1 are just given for indication (on a laptop with 3.5Gb memory, Intel®Core™2 2.7Ghz processor).

Limitations – Since feature extraction and smooth surface reconstruction are run independently, the extracted feature graph may have a certain discrepancy with respect to the smooth reconstructed surface. The feature preserving mesh generation algorithm yields a surface mesh with a faithful representation of sharp features provided that the protecting balls cover the gap between the extracted features and the reconstructed smooth surface. This condition puts one more constraint on the parameter δ and, hence on the mesh accuracy and sizing. Figure 7.18 illustrates what happens when the protecting balls do not cover the gap between the implicit

surface and the extracted polylines.

The pipeline currently described does not support adaptive sampling and the generation of meshes with non uniform sizing field. This however is not a real limitation of the method but simply results from our concern to limit and simplify the parameters of the algorithm.

7.6 Conclusion

In this chapter, we have presented an efficient and robust feature preserving mesh generation strategy from 3D point clouds. The approach we use builds a bridge between implicit surface reconstruction and mesh generation.

The method first extracts from the input point cloud a set of feature polylines through a feature detection process based on the covariance matrices of Voronoi cells. In parallel our algorithm runs on the input point cloud a reconstruction process, providing an implicit surface. A feature preserving variant of a Delaunay refinement process is then used to generate a mesh approximating the implicit surface and containing a faithful representation of the extracted sharp edges. The presented algorithm was implemented and successfully applied on various datasets including laser scanner datasets to generate meshes with enhanced trade-off between accuracy and size, even when sparse and noisy 3D point clouds were inputed.

As for future work, it would be interesting to apply our method to more challenging datasets provided by multi-view stereo reconstruction. We also plan to make our code available in the point set processing package of CGAL [Alliez et al., 2010b] as it might be useful in the fields which require simplified and realistic geometric models, like in urban modeling.

7.7 Publication

This work has been published in the SGP conference [Salman et al., 2010],

- N. Salman, M. Yvinec and Q. Mérigot. *Feature preserving mesh generation from 3D point clouds*. In Proceedings of the Eurographics Symposium on Geometry Processing, Lyon, France, July 2010.

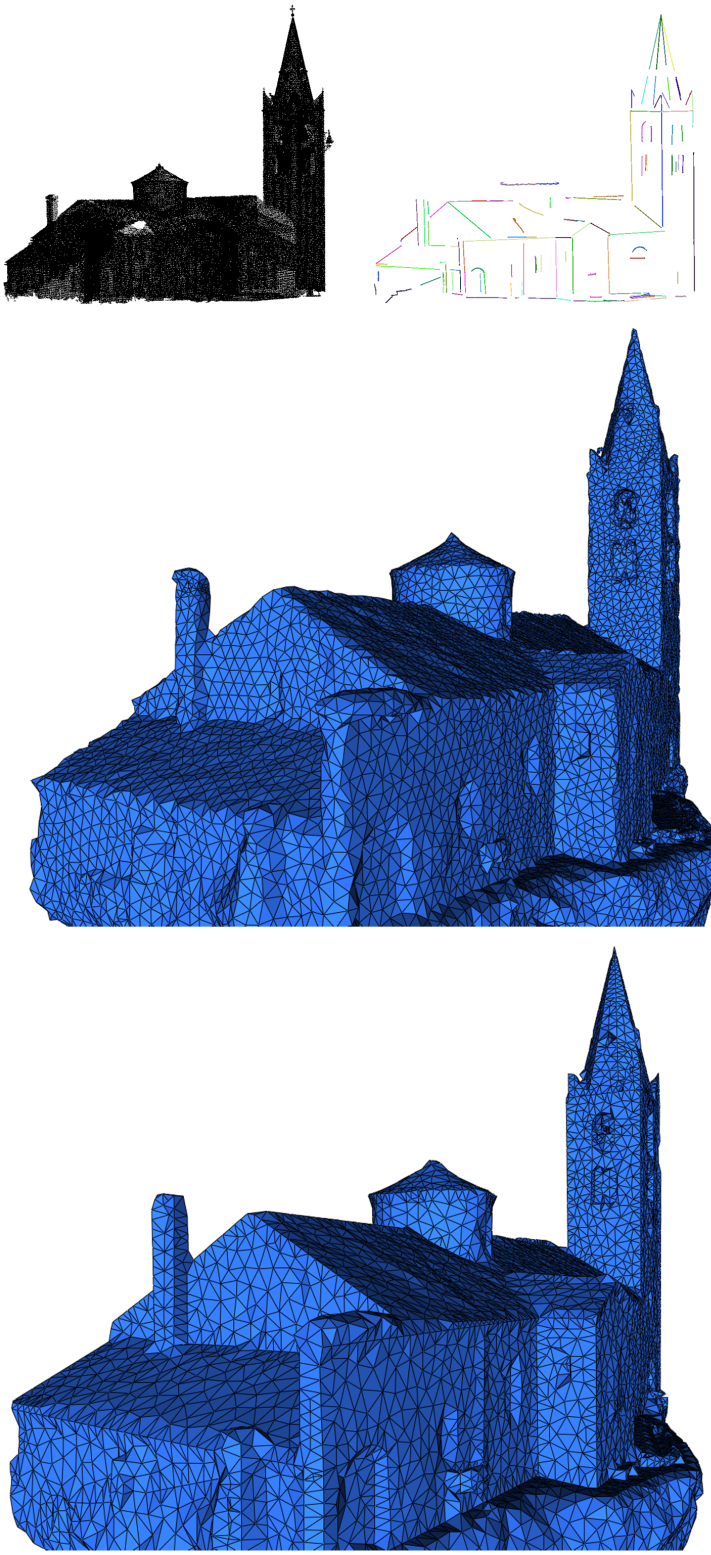


Figure 7.17: From top to bottom, left to right: input point cloud of *Church* model (*courtesy of INPG*) down-sampled to about 7.2% of the original dataset; extracted feature polylines; meshed Poisson implicit function; output of our feature preserving mesh generation (18k facets).

Model/ #points	Parameters t_1, t_2, δ	$\lvert\mathcal{F}\rvert$	Timings [*sec*] (1)	(2)	(3)	(4)	Fig.
fandisk/200k	50, 6, 3.5	2k	153	17	20	24	7.10
	50, 6, 7	2k	153	17	20	7	7.10
noise 1%	50, 6, 7	3k	126	32	41	7	7.15
octa-flower/167k	25, 4, 3.5	3k	96	19	53	13	7.12
carved_object/30k	16, 4, 4	2k	18	13	18	12	7.12
blade/50k	100, 7, 2.5	4k	34	52	32	9	7.12
block/40k	8, 2, 6	2k	16	20	37	7	7.14
Ramses/210k	150, 7, 5	8k	108	112	67	17	7.16
Lans church/500k	90, 10, 5	17k	368	306	184	78	7.17

Table 7.1: Statistics of our prototype implementation. The timings are given for each of the 4 stages in our method: sharp edges points detection (1); clustering (2); polyline construction and junction (3); and feature preserving mesh generation (4).

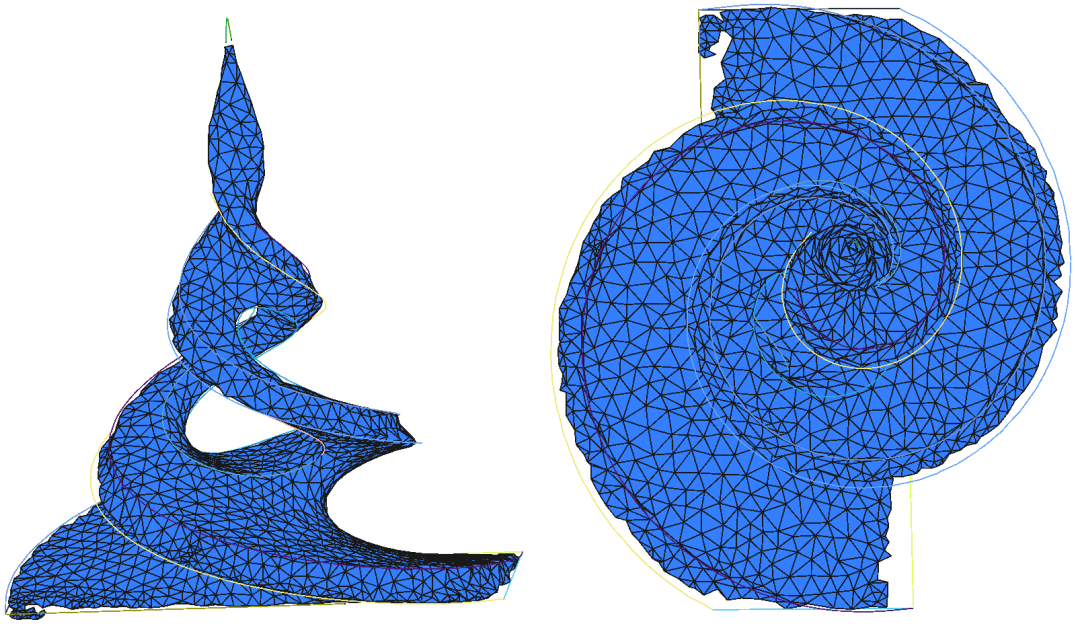


Figure 7.18: Two views of the output of our feature preserving mesh generation on the *twirl* model when protecting balls are not big enough to cover the gap between the implicit function and the extracted polylines.

Without continual growth and progress, such words as improvement, achievement, and success have no meaning.

Benjamin Franklin (1706-1790)

8

Conclusion

8.1 Summary

In this thesis we have addressed the problem of reconstruction of high quality 3D meshes from a set of scattered 3D points acquired from both multi-view passive stereo and laser range scanners. Two cases were considered.

First, we have proposed a method to reconstruct a 3D scene from an unstructured point cloud produced by a typical multi-view passive stereo pipeline and the calibrated images. The approach we use builds a bridge between multi-view stereo reconstruction algorithms and methods used in surface meshing. Particularly, our approach combines depth maps construction in the image planes with surface reconstruction through restricted Delaunay triangulation. A triangular depth map is built in each image plane respecting the main contours of the image. The triangles of these depth maps are then lifted into 3D space forming a triangle soup. The similarity of the triangle soup with the surface of the scene is enforced through visibility and photo-consistency constraints. A surface mesh is then generated using a Delaunay refinement surface mesh generation strategy. The accuracy of the presented method has been tested on numerous large-scale outdoor scenes. Our results show that the proposed method compares favorably with the current state of the art. Also, we stress the fact that some of the tools we have produced while building this algorithm are general and are not necessarily restricted to surface reconstruction but rather to process point clouds.

In the second approach, we have presented an efficient and robust feature preserving mesh generation strategy from 3D point clouds. The approach we developed combines implicit surface reconstruction and mesh generation. We start by extracting from the input point cloud a set of feature polylines through a robust feature detection process based on the covariance matrices of Voronoi cells. Also we follow the current trend in surface reconstruction, and run, in parallel, on the input point cloud an implicit reconstruction process, providing an implicit surface. A feature preserving variant of a Delaunay refinement process is then used to generate a mesh approximating the implicit surface and containing a faithful representation of the extracted sharp edges. The presented algorithm was implemented and successfully applied on various datasets including indoor and outdoor laser scanner datasets to generate meshes with enhanced trade-off between accuracy and size, even when sparse and noisy 3D point clouds were inputed.

8.2 Limitations and perspectives

Both aforementioned contributions can benefit from further enhancements. Let us mention a few of them that are of interest to us.

In our first contribution presented in Chapter 6, during the triangle filtering stage, it is sufficient for a triangle to not satisfy one of the mentioned constraints, to discard it from the triangle soup. As a result, the final triangle soup can contain holes leading to 3D meshes that are incomplete. To overcome this limitation, one solution would be to consider using a hole-filling heuristic on the resulting mesh. One common approach is to triangulate each connected component of the surfaces boundary, thus filling each hole with a patch that has the topology of a disc. We however would rather consider taking inspiration in the recent work of [Toldo & Fusiello, 2010] on gap-filling and try to apply it to our filtered triangle soup.

Being conservative at the reconstruction stage of our pipeline comes at the price that our reconstructed models are limited to the details captured by the filtered triangle soup. At the intuitive level, the Delaunay refinement procedure we employ is coupled with a sensing algorithm (oracle) probing the filtered triangle soup defined from the pre-processed point cloud. To obtain more accurate and visually pleasing results we would like to consider a final variational refinement to be applied to the generated meshes as a lightweight post-processing, by taking inspiration in the work of [Vu et al., 2009]. Additionally, incorporating in the reconstruction process a scale dependent sharp edges detection and recovery procedure to build geometrically simplified models is also expected.

In our second contribution presented in Chapter 7, since feature extraction and smooth surface reconstruction are run independently, the extracted approximate feature graph may have a certain discrepancy with respect to the smooth reconstructed surface. The feature preserving mesh generation algorithm yields a surface mesh with a faithful representation of sharp features provided that the protecting balls cover the *gap* between the extracted features and the reconstructed smooth surface. In the current implementation, the user provides a constant scalar field used as an upper bound for the distance between two protecting ball centers that are consecutive on a sharp edge. In practice, the user has to choose large enough protecting balls to cover the mentioned gap, as the current implementation can only refine balls. Said differently, the user has to intuit or guess the size of the largest gap to choose a correct starting size of protecting balls. For this reason, it would be more "flexible" to adapt the size of the protecting balls more locally. One straightforward way of obtaining gap-adaptive protecting balls would be to replace the current implementation with a process that works as follows: start by sampling the protecting balls in a first stage as described in Section 7.3.2 given the user defined balls radius, then, for each polyline grow the size of the balls that do not cover the gap between the extracted polyline and the implicit surface to cover it and make sure that the sampling conditions [Cheng et al., 2007b] remain locally satisfied. This might seem simple at first sight, but we expect multiple particular cases to arise especially in the vicinity of polylines junction points.

We think, it would be interesting, for the Gyroviz project, to evaluate our feature extraction process on fairly simple real-world buildings for which the number of sharp edges can be enumerated *a priori* once the observation scale is fixed and compare to the number of features extracted. The number of extracted features would be computed by measuring the percentage of ground truth sharp features that are within a distance smaller than a certain threshold to the extracted sharp features by our algorithm. We think that extracting 80% of the sharp features would be satisfactory for reconstructing urban scenes in which buildings are entirely framed by the camera.

Computing feature preserving meshes from point clouds is a topic of high practical relevance. Although our method showed to behave robustly in the experiments we conducted, we think that a more in depth quantitative evaluation and clear comparison to other alternative approaches, for instance [Daniels et al., 2007, Jenke et al., 2008], has to be done. This raises the more interesting question that is: why not establishing an online public benchmark conceptually similar to [Seitz et al., 2006, Strecha et al., 2008], that would evaluate feature extraction and mesh generation algorithms from point clouds? One possibility would be to provide *new* 3D point clouds constructed from mesh models. The meshes will serve as the geometrical ground truth to evaluate the quality of: the extracted sharp features for the algorithms limited to feature extraction, and the feature preserving meshes for others. The evaluation set would include for instance, the distance from the practitioner's extracted approximate feature graph to the nearest ground truth sharp features, the percentage of extracted sharp features and an approximation of the Hausdorff distance between the generated meshes and the ground truth models. Algorithms would then be compared relative to these tests.

On a smaller scale, we plan to exploit the Delaunay refinement mesh generation algorithm flexibility to apply our method to more challenging datasets. Although the method was found to behave robustly in the experiments we conducted in the presence of noise, the CVCM feature detection technique we employ is only robust to "some outliers" [Merigot et al., 2009]. This limitation is particularly visible when we try to extract sharp features from the datasets, used to test our first contribution, provided by multi-view passive stereo acquisition. The feature extraction step produces some intractable results as illustrated on the left side of Figure 8.1. To help our feature extraction step working with these datasets, entangled with noise and outliers, one solution would be to extend our second contribution with the filtering heuristics described in Section 6.3.1. Applying a filtering stage is only a first attempt to solve this hard problem. The preliminary results we got are encouraging as illustrated on the right side of Figure 8.1.

Finally, we are working to make our code available and we hope that it will be useful in the fields which require simplified and realistic geometric models.

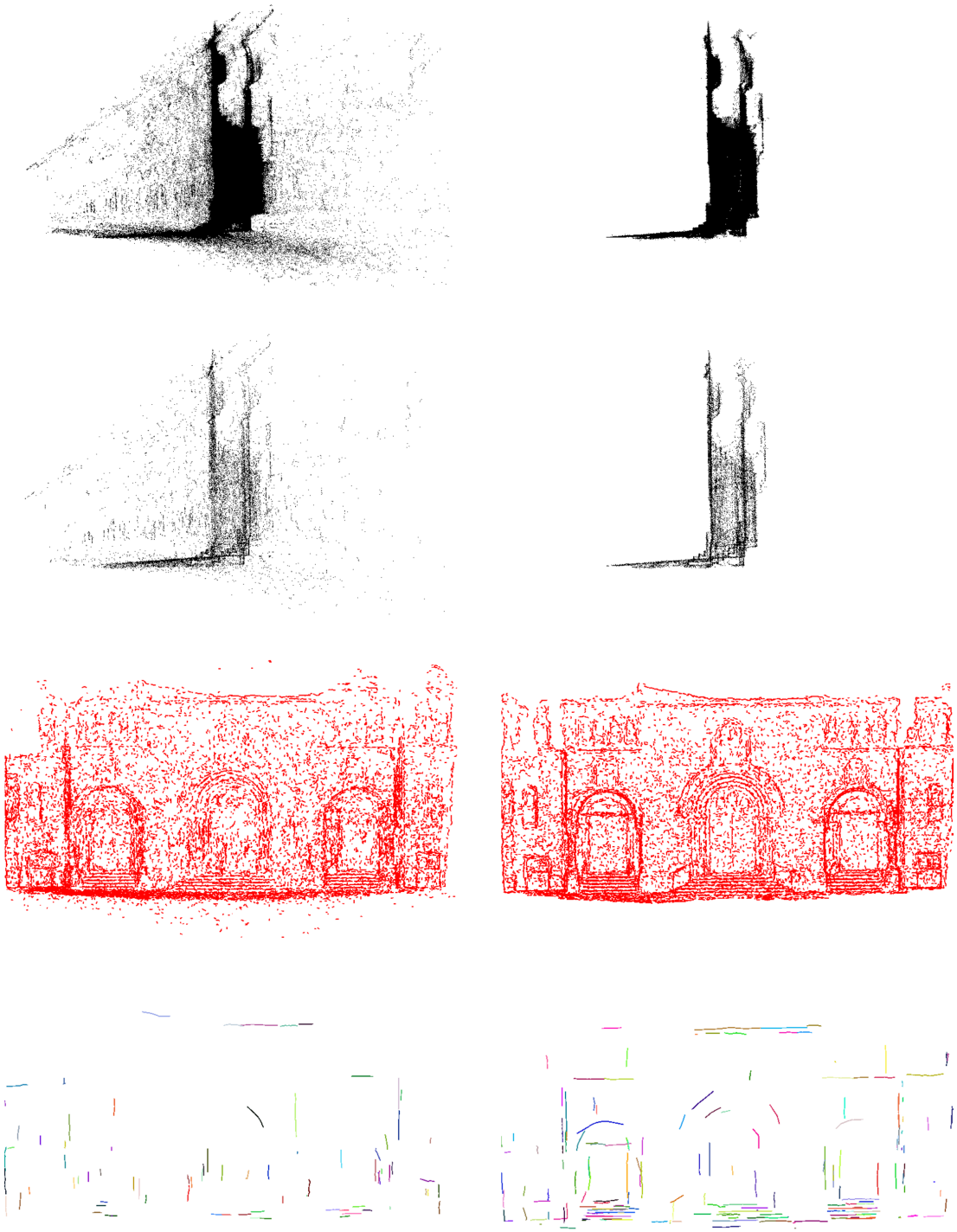


Figure 8.1: From top to bottom: side view of the input point cloud from the *Herz-Jesu-p25* dataset [Strecha et al., 2008]; side view of the identified edge points; front view of the associated feature directions; and the extracted polylines. Left column: unprocessed point cloud. Right column: inliers.

C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do, it blows away your whole leg.

Bjarne Stroustrup

A

Point set processing in CGAL

Surface reconstruction from point sets is often a sequential process with the following steps: point set acquisition; outliers removal; simplification to reduce the number of input points; smoothing to reduce noise in the input data; normal estimation and orientation when the normals are not already provided by the acquisition step; and surface reconstruction. In what follows, we describe the algorithms we developed, in collaboration with Pierre Alliez and Laurent Saboret for the Gyroviz project, to pre-process the point sets before surface reconstruction. Note that surface reconstruction algorithms were also integrated to CGAL in [Alliez et al., 2010a] for the Gyroviz Project.

A.1 Introduction

This CGAL component implements methods to analyze and process unorganized 3D point sets. The input is an unorganized 3D point set, possibly with normal attributes (un-oriented or oriented). This point set can be analyzed to measure geometric properties such as average spacing between the points and their K nearest neighbors. It can be processed with functions devoted to the simplification, outliers removal, smoothing, normal estimation and normal orientation. The processing of point sets is often needed in applications dealing with measurement data, such as surface reconstruction from laser scanned data (see Figure A.1).

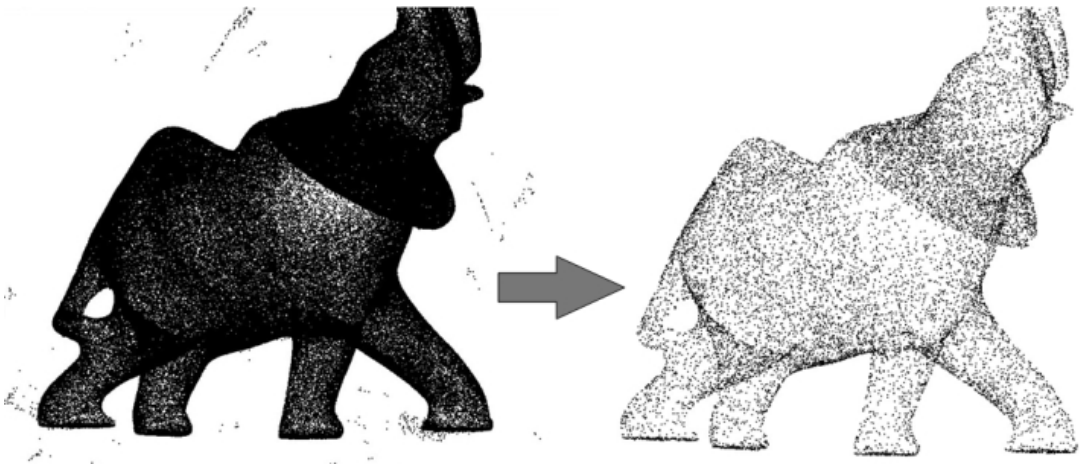


Figure A.1: Point set processing. Left: 275k points sampled on the statue of an elephant with a Minolta laser scanner. Right: point set after clean-up and simplification to 17k points.

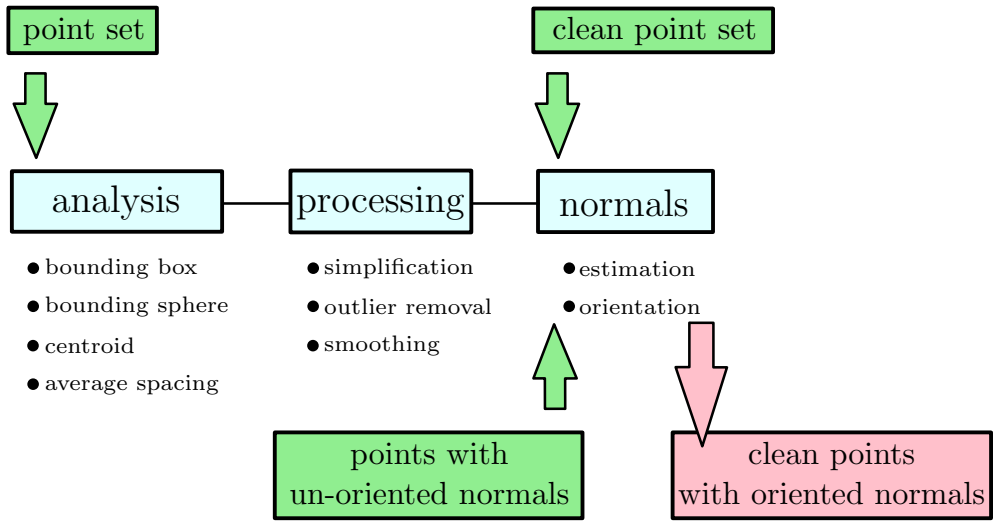


Figure A.2: Point set processing pipeline for surface reconstruction. The algorithms listed in gray are available from other CGAL components (bounding volumes and principal component analysis).

In the context of surface reconstruction we can position the elements of this component along the common surface reconstruction pipeline (Figure A.2) which involves the following steps:

1. scanning and scan alignment to produce a set of points or points with normals (alignment is not yet covered in CGAL);
2. outliers removal;
3. simplification to reduce the number of input points;
4. smoothing to reduce noise in the input data;
5. normal estimation and orientation when the normals are not already provided by the acquisition device;
6. surface reconstruction [Alliez et al., 2010a] deals with surface reconstruction from point sets with normal attributes.

A.2 Input/Output

A.2.1 Property Maps

The algorithms of this component take as input parameters iterator ranges of 3D points, or of 3D points with normals. The property maps are used to access the point or normal information from the input data, so as to let the user decide upon the implementation of a point with normal. The latter can be represented as, e.g., a class derived from the CGAL 3D point, or as a `std::pair<Point_3<K>, Vector_3<K> >`, or as a `boost::tuple<..., Point_3<K>, ..., Vector_3<K> >`.

Another component provides property maps to support these cases:

- `CGAL::Dereference_property_map<T>`;
- `CGAL::First_of_pair_property_map<Pair>` and `CGAL::Second_of_pair_property_map<Pair>`;
- `CGAL::Nth_of_tuple_property_map<N, Tuple>`.

Where `CGAL::Dereference_property_map<Point_3>` is the default value of the position property map expected by all functions in this component. See below examples using pair and tuple property maps.

Note: Users of this package may use other types to represent positions and normals if they implement the corresponding property maps.

A.2.2 Streams

We provide functions to read and write sets of points or sets of points with normals from the following ASCII file formats:

- XYZ (three point coordinates `x y z` per line or three point coordinates and three normal vector coordinates `x y z nx ny nz` per line);
- OFF (Object File Format) [Phillips et al., 1993].

`CGAL::read_xyz_points`
`CGAL::read_off_points`
`CGAL::write_off_points`
`CGAL::write_xyz_points`

A.2.3 Example

The following example reads a point set from an input file and writes it to a file, both in the XYZ format. Positions and normals are stored in pairs and accessed through property maps.

```
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/property_map.h>
#include <CGAL/IO/read_xyz_points.h>
#include <CGAL/IO/write_xyz_points.h>

#include <utility> // defines std::pair
#include <vector>
#include <fstream>

// types
typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef Kernel::Point_3 Point;
typedef Kernel::Vector_3 Vector;

// Point with normal vector stored as a std::pair.
typedef std::pair<Point, Vector> Pwn;

int main(void)
{
```

```
    // Reads a .xyz point set file in points[].
    // Note: read_xyz_points_and_normals() requires an output iterator
    // over points and as well as property maps to access each
    // point position and normal.
    std::vector<Pwn> points;
    std::ifstream in("data/oni.xyz");
    if (!in ||
        !CGAL::read_xyz_points_and_normals(
            in,std::back_inserter(points),
            CGAL::First_of_pair_property_map<Pwn>(),
            CGAL::Second_of_pair_property_map<Pwn>()))
    {
      std::cerr << "Error: cannot read file data/oni.xyz" << std::endl;
      return EXIT_FAILURE;
    }

    // Saves point set.
    // Note: write_xyz_points_and_normals() requires an output iterator
    // over points as well as property maps to access each
    // point position and normal.
    std::ofstream out("oni_copy.xyz");
    if (!out ||
        !CGAL::write_xyz_points_and_normals(
            out, points.begin(), points.end(),
            CGAL::First_of_pair_property_map<Pwn>(),
            CGAL::Second_of_pair_property_map<Pwn>()))
    {
      return EXIT_FAILURE;
    }

    return EXIT_SUCCESS;
}
```

A.3 Analysis

Function CGAL::compute_average_spacing() computes the average spacing of all input points to their k nearest neighbor points, k being pecified by the user. As it provides an order of a point set density, this function is used downstream the surface reconstruction pipeline to automatically determine some parameters such as output mesh sizing for surface reconstruction.

A.3.1 Example

The following example reads a point set in the XYZ format and computes the average spacing. Index, position and color are stored in a tuple and accessed through property maps.

```
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/compute_average_spacing.h>
#include <CGAL/IO/read_xyz_points.h>

#include <vector>
#include <fstream>
```

```
7  #include <boost/tuple/tuple.hpp>
8
9  // Types
10 typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
11 typedef Kernel::FT FT;
12 typedef Kernel::Point_3 Point;
13
14 // Data type := index, followed by the point, followed by three integers
15 // that define the Red Green Blue color of the point.
16 typedef boost::tuple<int, Point, int, int, int> IndexPointWithColorTuple;
17
18 int main(void)
19 {
20   // Reads a .xyz point set file in points.
21   // As the point is the second element of the tuple (with index 1)
22   // we use a property map that accesses the 1st element of the tuple.
23   std::vector<IndexPointWithColorTuple> points;
24   std::ifstream stream("data/sphere_20k.xyz");
25   if (!stream ||
26       !CGAL::read_xyz_points(
27           stream, std::back_inserter(points),
28           CGAL::Nth_of_tuple_property_map<1,IndexPointWithColorTuple>()))
29   {
30     std::cerr << "Error: cannot read file data/sphere_20k.xyz"
31         << std::endl;
32     return EXIT_FAILURE;
33   }
34
35   // Initialize index and RGB color fields in tuple.
36   // As the index and RGB color are respectively the first and
37   // third-fifth elements of the tuple we use a get function
38   // from the property map that accesses the 0 and 2-4th elements
39   // of the tuple.
40     for(unsigned int i = 0; i < points.size(); i++)
41     {
42       points[i].get<0>() = i;   // set index value of tuple to i
43
44       points[i].get<2>() = 0;   // set RGB color to black
45       points[i].get<3>() = 0;
46       points[i].get<4>() = 0;
47     }
48
49     // Computes average spacing.
50     const unsigned int nb_neighbors = 6; // 1 ring
51     FT average_spacing = CGAL::compute_average_spacing(
52                          points.begin(), points.end(),
53                          CGAL::Nth_of_tuple_property_map<1,
54               IndexPointWithColorTuple>(),
55                          nb_neighbors);
56     std::cout << "Average spacing: " << average_spacing << std::endl;
57
58     return EXIT_SUCCESS;
59 }
```

Note: Other functions such as centroid or bounding volumes are found in other CGAL components: `CGAL::centroid`, `CGAL::bounding_box` and `CGAL::bounding_sphere`.

A.4 Outliers Removal

Function `CGAL::remove_outliers()` deletes a user-specified fraction of outliers from an input point set. More specifically, it sorts the input points in increasing order of average squared distances to their k nearest neighbors and deletes the points with largest value.

A.4.1 Example

The following example reads a point set and eliminates 5% of its points. It uses the default position property map `CGAL::Dereference_property_map<Point_3>` (it is the default position property map of all functions in this CGAL component).

```
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/property_map.h>
#include <CGAL/remove_outliers.h>
#include <CGAL/IO/read_xyz_points.h>

#include <vector>
#include <fstream>

// types
typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef Kernel::Point_3 Pt;

int main(void)
{
  // Reads a .xyz point set file in points[].
  // The Dereference_property_map property map can be omitted here as
  // it is the default value.
  std::vector<Pt> points;
  std::ifstream stream("data/oni.xyz");
  if (!stream ||
      !CGAL::read_xyz_points(stream, std::back_inserter(points),
                             CGAL::Dereference_property_map<Pt>()))
  {
    std::cerr << "Error: cannot read file data/oni.xyz" << std::endl;
    return EXIT_FAILURE;
  }

  // Removes outliers using erase-remove idiom.
  // The Dereference_property_map property map can be omitted here as
  // it is the default value.
  const double removed_percentage = 5.0; // % of points to remove
  const int nb_neighbors = 24; // considers 24 nearest neighbor points
  points.erase(CGAL::remove_outliers(points.begin(), points.end(),
                                     CGAL::Dereference_property_map<Pt>(),
                                     nb_neighbors, removed_percentage),
               points.end());
```

```
37
38    // Optional: after erase(), use Scott Meyer's "swap trick" to trim
39    //           excess capacity
40    std::vector<Pt>(points).swap(points);
41
42    return EXIT_SUCCESS;
43  }
```

A.5 Simplification

Two simplification functions are devised to reduce an input point set, either randomly or using a grid-based clustering approach.

Function CGAL::random_simplify_point_set() randomly deletes a user-specified fraction of points from the input point set. This algorithm is fast.

Function CGAL::grid_simplify_point_set() considers a regular grid covering the bounding box of the input point set, and clusters all points sharing the same cell of the grid by picking as representant one arbitrarily chosen point. However, the grid simplification algorithm is slower than CGAL::random_simplify_point_set().

A.5.1 Example

The following example reads a point set and simplifies it by clustering.

```
1  #include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
2  #include <CGAL/grid_simplify_point_set.h>
3  #include <CGAL/IO/read_xyz_points.h>
4
5  #include <vector>
6  #include <fstream>
7
8  // types
9  typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
10 typedef Kernel::Point_3 Point;
11
12 int main(void)
13 {
14   // Reads a .xyz point set file in points[].
15   std::vector<Point> points;
16   std::ifstream stream("data/oni.xyz");
17   if (!stream ||
18       !CGAL::read_xyz_points(stream, std::back_inserter(points)))
19   {
20     std::cerr << "Error: cannot read file data/oni.xyz" << std::endl;
21     return EXIT_FAILURE;
22   }
23
24   // simplification by clustering using erase-remove idiom
25   double cell_size = 0.001;
26   points.erase(CGAL::grid_simplify_point_set(points.begin(),
27          points.end(), cell_size), points.end());
```

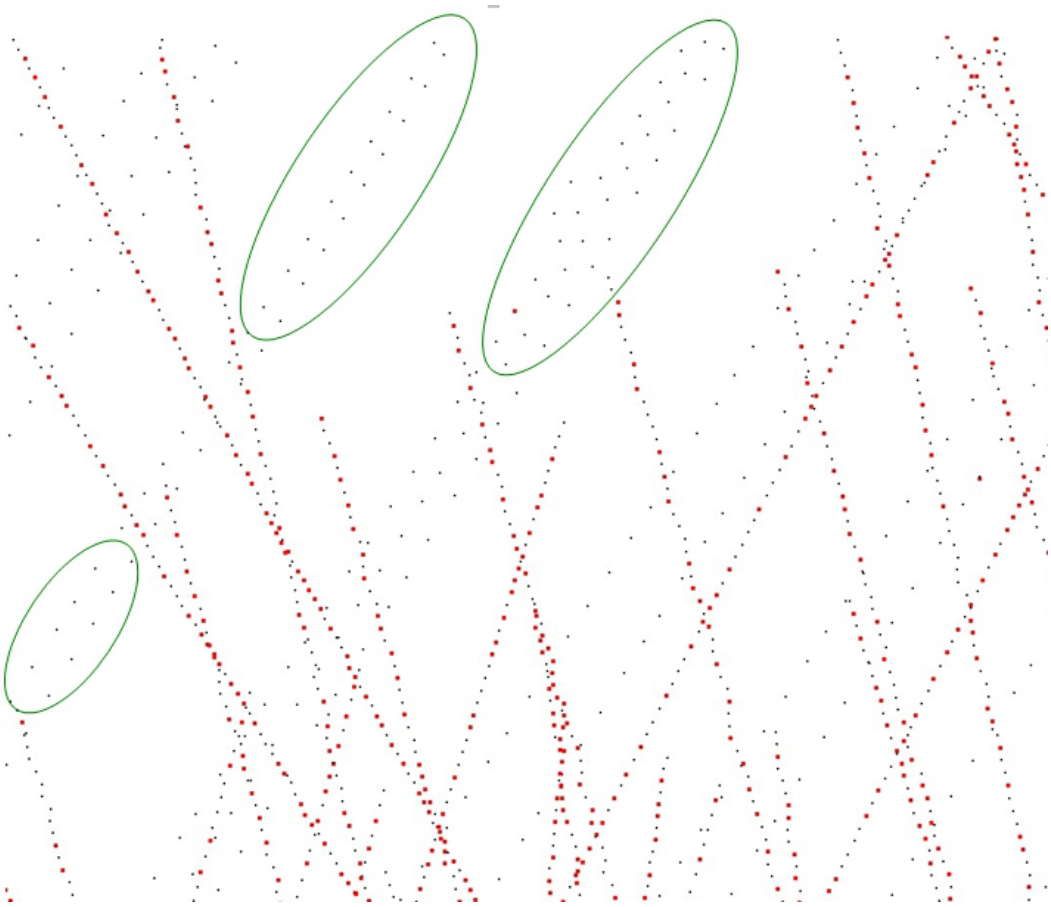


Figure A.3: Point set simplification through grid-based clustering. Removed points are depicted in red. Notice how low-density areas (in green) are not simplified.

```
28
29    // Optional: after erase(), use Scott Meyer's "swap trick" to trim
30    //           excess capacity
31    std::vector<Point>(points).swap(points);
32
33    return EXIT_SUCCESS;
34  }
```

A.6 Smoothing

Function CGAL::jet_smooth_point_set smooths the input point set by projecting each point onto a smooth parametric surface patch (so-called jet surface) fitted over its k nearest neighbors. For a more detailed description of the method please refer to Section 6.3.1.

A.6.1 Example

The following example generates a set of nine points close to the xy plane and smooths them using eight nearest neighbors:

```
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/jet_smooth_point_set.h>
#include <vector>

// types
typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef Kernel::Point_3 Point;

int main(void)
{
  // generate point set
  std::vector<Point> points;
  points.push_back(Point( 0.0, 0.0, 0.001));
  points.push_back(Point(-0.1,-0.1, 0.002));
  points.push_back(Point(-0.1,-0.2, 0.001));
  points.push_back(Point(-0.1, 0.1, 0.002));
  points.push_back(Point( 0.1,-0.1, 0.000));
  points.push_back(Point( 0.1, 0.2, 0.001));
  points.push_back(Point( 0.2, 0.0, 0.002));
  points.push_back(Point( 0.2, 0.1, 0.000));
  points.push_back(Point( 0.0,-0.1, 0.001));

  // Smoothing.
  const unsigned int nb_neighbors = 8; // default is 24 for real-world
                     // point sets
  CGAL::jet_smooth_point_set(points.begin(), points.end(), nb_neighbors);

  return EXIT_SUCCESS;
}
```

A.7 Normal Estimation

Assuming a point set sampled over an inferred surface $\mathcal{S}$, two functions provide an estimate of the normal to $\mathcal{S}$ at each point. The result is an un-oriented normal vector for each input point.

Function CGAL::jet_estimate_normals() estimates the normal direction at each point from the input set by fitting a jet surface over its k nearest neighbors. The default jet is a quadric surface. This algorithm is well suited to point sets scattered over curved surfaces.

Function CGAL::pca_estimate_normals() estimates the normal direction at each point from the set by linear least squares fitting of a plane over its k nearest neighbors. This algorithm is simpler and faster than CGAL::jet_estimate_normals().

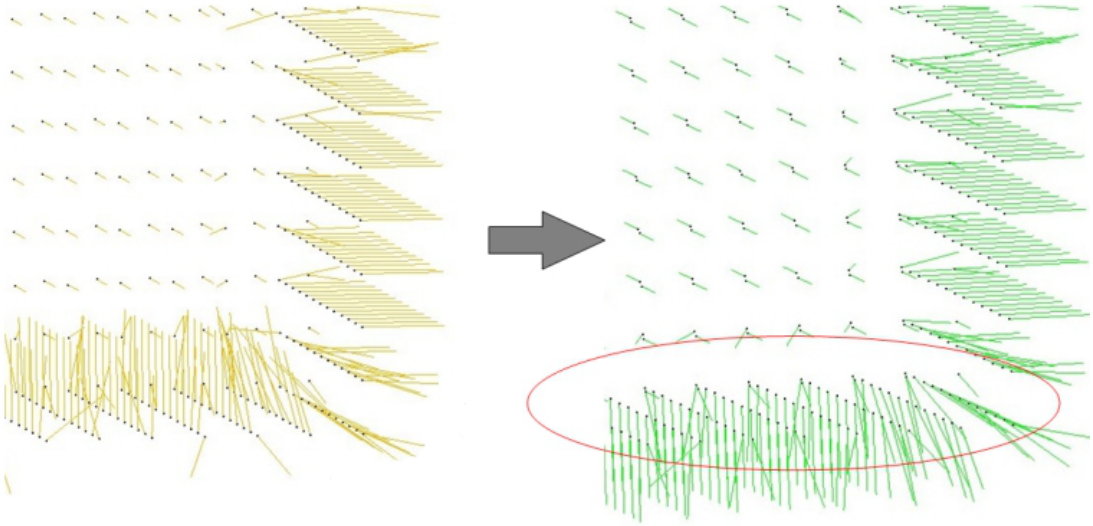


Figure A.4: Normal orientation of a sampled cube surface. Left: un-oriented normals. Right: orientation of right face normals is propagated to bottom face.

A.8 Normal Orientation

Function `CGAL::mst_orient_normals()` orients the normals of a set of points with un-oriented normals using the method described by [Hoppe et al., 1992] in *Surface reconstruction from unorganized points*. More specifically, this method constructs a Riemannian graph over the input points (the graph of the k nearest neighbor points). Each edge between two points of this graph is assigned a weight measuring the discrepancy for propagating the normal orientation along this edge. We follow Hoppe *et al.* and use the absolute value of the dot product of two linked normals as a measure of normal variation. The normal orientation is then propagated along the minimum spanning tree computed of this graph starting at a seed point with fixed normal. The result is an oriented normal vector for each input un-oriented normal, except for the normals which could not be successfully oriented.

A.8.1 Example

The following example reads a point set from a file, estimates the normals through PCA over the six nearest neighbors and orients the normals:

```
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/pca_estimate_normals.h>
#include <CGAL/mst_orient_normals.h>
#include <CGAL/property_map.h>
#include <CGAL/IO/read_xyz_points.h>

#include <utility> // defines std::pair
#include <list>
#include <fstream>

// Types
```

```
typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef Kernel::Point_3 Point;
typedef Kernel::Vector_3 Vector;

// Point with normal vector stored in a std::pair.
typedef std::pair<Point, Vector> PtVecPair;

int main(void)
{
 // Reads a .xyz point set file in points[].
 std::list<PtVecPair> points;
 std::ifstream stream("data/sphere_20k.xyz");
 if (!stream ||
     !CGAL::read_xyz_points(stream,
                            std::back_inserter(points),
                            CGAL::First_of_pair_property_map<PtVecPair>()))
 {
     std::cerr << "Error: cannot read file data/sphere_20k.xyz"
         << std::endl;
     return EXIT_FAILURE;
 }

 // Estimates normals direction.
 // Note: pca_estimate_normals() requires an iterator over points
 // as well as property maps to access each point's position and normal.
 const int nb_neighbors = 18; // K-nearest neighbors = 3 rings
 CGAL::pca_estimate_normals(points.begin(), points.end(),
                            CGAL::First_of_pair_property_map<PtVecPair>(),
                            CGAL::Second_of_pair_property_map<PtVecPair>(),
                            nb_neighbors);

 // Orients normals.
 // Note: mst_orient_normals() requires an iterator over points
 // as well as property maps to access each point's position and normal.
 std::list<PtVecPair>::iterator unoriented_points_begin =
  CGAL::mst_orient_normals(points.begin(), points.end(),
                           CGAL::First_of_pair_property_map<PtVecPair>(),
                           CGAL::Second_of_pair_property_map<PtVecPair>(),
                           nb_neighbors);

   // Optional: delete points with an unoriented normal
   //         if you plan to call a reconstruction
   //           algorithm that expects oriented normals.
   points.erase(unoriented_points_begin, points.end());

   // Optional: after erase(), use Scott Meyer's "swap trick"
   //           to trim excess capacity
   std::list<PtVecPair>(points).swap(points);

   return EXIT_SUCCESS;
}
```

Bibliography

[aas, 2006] (2006). AIM@SHAPE. http://shapes.aimatshape.net. 16

[cga, 2010] (2010). CGAL, Computational Geometry Algorithms Library. http://www.cgal.org. 61, 69, 91

[cim, 2010] (2010). CIMG, C++ Template Image Processing Toolkit. http://cimg.sourceforge.net. 69

[Adamson & Alexa, 2003] Adamson, A. & Alexa, M. (2003). Approximating and intersecting surfaces from points. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (pp. 239).: Eurographics Association. 40

[Adamson & Alexa, 2006] Adamson, A. & Alexa, M. (2006). Anisotropic point set surfaces. In *Proc. of Afrigraph '06* (pp.13). 78

[Adamy et al., 2002] Adamy, U., Giesen, J., & John, M. (2002). Surface reconstruction using umbrella filters. *Computational Geometry*, 21(1-2), 63–86. 33

[Agarwal et al., 2009] Agarwal, S., Snavely, N., Simon, I., Seitz, S., & Szeliski, R. (2009). Building rome in a day. In *Proc. 12th Int. Conf. on Computer Vision, Kyoto, Japan*: Citeseer. 24

[Alexa et al., 2003] Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., & Silva, C. (2003). Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, (pp. 3–15). 39

[Alliez et al., 2007] Alliez, P., Cohen-Steiner, D., Tong, Y., & Desbrun, M. (2007). Voronoi-based variational reconstruction of unoriented point sets. In *Proc. of SGP '07* (pp. 39–48). 45, 83, 93

[Alliez et al., 2010a] Alliez, P., Saboret, L., & Guennebaud, G. (2010a). Surface reconstruction from point sets. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.6 edition. 107, 108

[Alliez et al., 2010b] Alliez, P., Saboret, L., & Salman, N. (2010b). Point set processing. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.6 edition. 61, 69, 76, 93, 99

[Alliez et al., 2010c] Alliez, P., Tayeb, S., & Wormser, C. (2010c). Aabb tree. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.6 edition. 69

[Amenta, 1999] Amenta, N. (1999). The crust algorithm for 3d surface reconstruction. In *Proceedings of the fifteenth annual symposium on Computational geometry* (pp. 424).: ACM. 33

[Amenta & Bern, 1998] Amenta, N. & Bern, M. (1998). Surface reconstruction by voronoi filtering. In *Proceedings of the fourteenth annual symposium on Computational geometry* (pp.48).: ACM. 11, 30, 33, 58

[Amenta et al., 2000] Amenta, N., Choi, S., Dey, T., & Leekha, N. (2000). A simple algorithm for homeomorphic surface reconstruction. In *Proceedings of the sixteenth annual symposium on Computational geometry* (pp. 222).: ACM. 13, 34, 35

[Amenta et al., 2001] Amenta, N., Choi, S., & Kolluri, R. K. (2001). The power crust. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications* (pp. 249–266). New York, NY, USA: ACM. 34

[Attali et al., 2009] Attali, D., Boissonnat, J., & Edelsbrunner, H. (2009). Stability and computation of medial axes-a state-of-the-art report. *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, (pp. 109–125). 11

[Attali et al., 2003] Attali, D., Boissonnat, J., & Lieutier, A. (2003). Complexity of the delaunay triangulation of points on surfaces the smooth case. In *Proceedings of the nineteenth annual symposium on Computational geometry* (pp. 210).: ACM. 8

[Attene & Spagnuolo, 2000] Attene, M. & Spagnuolo, M. (2000). Automatic surface reconstruction from point sets in space. In *Computer Graphics Forum*, volume 19 (pp. 457–465).: John Wiley & Sons. 30

[Baehmann et al., 1987] Baehmann, P., Wittchen, S., Shephard, M., Grice, K., & Yerry, M. (1987). Robust, geometrically based, automatic two-dimensional mesh generation. *International Journal for Numerical Methods in Engineering*, 24(6), 1043–1078. 48

[Baker et al., 1988] Baker, B., Grosse, E., & Rafferty, C. (1988). Nonobtuse triangulation of polygons. *Discrete and Computational Geometry*, 3(1), 147–168. 48

[Banno & Ikeuchi, 2005] Banno, A. & Ikeuchi, K. (2005). Shape recovery of 3d data obtained from a moving range sensor by using image sequences. 55

[Banno et al., 2008] Banno, A., Masuda, T., Oishi, T., & Ikeuchi, K. (2008). Flying laser range sensor for large-scale site-modeling and its applications in bayon digital archival project. *International Journal of Computer Vision*, 78(2), 207–222. 55

[Bentley, 1975] Bentley, J. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9), 517. 24

[Bern et al., 1990] Bern, M., Eppstein, D., & Gilbert, J. (1990). Provably good mesh generation. In *31st annual Symposium on Foundations of Computer Science: proceedings, October 22-24, 1990, St. Louis, Missouri* (pp. 231).: IEEE Computer Society Press. 48, 51

[Bern et al., 1994] Bern, M., Eppstein, D., & Gilbert, J. (1994). Provably good mesh generation. *Journal of Computer and System Sciences*, 48(3), 384–409. 51

[Bern & Plassmann, 2000] Bern, M. & Plassmann, P. (2000). Mesh generation. *Handbook of computational geometry*, (pp. 291–332). 47

[Bernardini et al., 1999] Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G., et al. (1999). The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4), 349–359. 18, 32

[Boehler et al., 2003] Boehler, W., Bordas Vicent, M., & Marbs, A. (2003). Investigating laser scanner accuracy. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34(Part 5), 696–701. 17

[Boissonnat & Cazals, 2001] Boissonnat, J. & Cazals, F. (2001). Coarse-to-fine surface simplification with geometric guarantees. In *Computer Graphics Forum*, volume 20 (pp. 490–499).: John Wiley & Sons. 38

[Boissonnat & Oudot, 2004] Boissonnat, J. & Oudot, S. (2004). An effective condition for sampling surfaces with guarantees. In *Proceedings of the ninth ACM symposium on Solid modeling and applications* (pp. 112).: Eurographics Association. 40

[Boissonnat, 1984] Boissonnat, J.-D. (1984). Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4). 30, 31

[Boissonnat & Cazals, 2000] Boissonnat, J.-D. & Cazals, F. (2000). Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proc. of SCG '00* (pp. 223–232). 37, 38

[Boissonnat et al., 2000] Boissonnat, J.-D., Devillers, O., Teillaud, M., & Yvinec, M. (2000). Triangulations in cgal (extended abstract). In *Proc. of SCG '00* (pp. 11–18). 69

[Boissonnat & Oudot, 2005] Boissonnat, J.-D. & Oudot, S. (2005). Provably good sampling and meshing of surfaces. *Graphical Models*. 13, 51, 52, 58, 59, 68, 80, 90, 91, 96

[Boissonnat & Yvinec, 1998] Boissonnat, J.-D. & Yvinec, M. (1998). *Algorithmic geometry*. New York, NY, USA: Cambridge University Press. 7, 8

[Bolitho et al., 2007] Bolitho, M., Kazhdan, M., Burns, R., & Hoppe, H. (2007). Multilevel streaming for out-of-core surface reconstruction. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing* (pp. 69–78). Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. 18

[Boltcheva et al., 2009] Boltcheva, D., Yvinec, M., & Boissonnat, J.-D. (2009). Feature preserving delaunay mesh generation from 3d multi-material images. *Computer Graphics Forum*, 28, 1455–14645. special issue for EUROGRAPHICS Symposium on Geometry Processing. 90

[Botsch et al., 2010] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., & Levy, B. (2010). *Polygon Mesh Processing*. AK Peters. 47

[Bowyer, 1981] Bowyer, A. (1981). Computing dirichlet tessellations. *The Computer Journal*, 24(2), 162. 50

[Boykov & Kolmogorov, 2003] Boykov, Y. & Kolmogorov, V. (2003). Computing geodesics and minimal surfaces via graph cuts. In *Proceedings of the Ninth IEEE International Conference on Computer Vision-Volume 2* (pp.26̃).: IEEE Computer Society. 43

[Bradley et al., 2008] Bradley, D., Boubekeur, T., Berlin, T., & Heidrich, W. (2008). Accurate multiview reconstruction using robust binocular stereo and surface meshing. In *Proc. of CVPR*: Citeseer. 45

[Callieri et al., 2009] Callieri, M., Cignoni, P., Dellepiane, M., & Scopigno, R. (2009). Pushing time-of-flight scanners to the limit. 16

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6), 679–698. 64, 65

[Carr et al., 2001] Carr, J., Beatson, R., Cherrie, J., Mitchell, T., Fright, W., McCallum, B., & Evans, T. (2001). Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (pp.76̃).: ACM. 41

[Cavendish et al., 1985] Cavendish, J., Field, D., & Frey, W. (1985). An apporach to automatic three-dimensional finite element mesh generation. *International Journal for Numerical Methods in Engineering*, 21(2), 329–347. 50

[Cazals & Giesen, 2004] Cazals, F. & Giesen, J. (2004). *Delaunay Triangulation Based Surface Reconstruction: Ideas and Algorithms*. Technical Report RR-5393, INRIA. 7

[Cazals & Giesen, 2006] Cazals, F. & Giesen, J. (2006). *Effective computational geometry for curves and surfaces*. Springer-Verlag Berlin Heidelberg. 29

[Cazals & Pouget, 2003] Cazals, F. & Pouget, M. (2003). Estimating differential quantities using polynomial fitting of osculating jets. In *Proc. of SGP '03*. 62

[Chaine, 2003] Chaine, R. (2003). A geometric convection approach of 3-d reconstruction. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (pp. 229).: Eurographics Association. 31

[Chen & Medioni, 1997] Chen, Q. & Medioni, G. (1997). Image synthesis from a sparse set of views. In *Proceedings of the 8th conference on Visualization'97* (pp. 269).: IEEE Computer Society Press. 67

[Cheng et al., 2007a] Cheng, S., Dey, T., & Levine, J. (2007a). A practical delaunay meshing algorithm for a large class of domains. In *Proc. of IMR '07* (pp. 477–494).: Springer. 51, 80, 82, 90

[Cheng et al., 2004] Cheng, S., Dey, T., Ramos, E., & Ray, T. (2004). Sampling and meshing a surface with guaranteed topology and geometry. In *Proceedings of the twentieth annual symposium on Computational geometry* (pp. 289).: ACM. 51

[Cheng & Poon, 2003] Cheng, S. & Poon, S. (2003). Graded conforming delaunay tetrahedralization with bounded radius-edge ratio. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 304).: Society for Industrial and Applied Mathematics. 51

[Cheng et al., 2007b] Cheng, S.-W., Dey, T. K., & Ramos, E. A. (2007b). Delaunay refinement for piecewise smooth complexes. In *Proc. of SODA '07* (pp. 1096–1105). 51, 52, 80, 90, 104

[Chew, 1989] Chew, L. (1989). Guaranteed-quality triangular meshes. 50

[Chew, 1993] Chew, L. (1993). Guaranteed-quality mesh generation for curved surfaces. In *Proceedings of the ninth annual symposium on Computational geometry* (pp. 280).: ACM. 13, 51

[Cignoni et al., 2008] Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., & Ranzuglia, G. (2008). Meshlab: an open-source mesh processing tool. 48

[Cohen-Steiner & Da, 2004] Cohen-Steiner, D. & Da, F. (2004). A greedy delaunay-based surface reconstruction algorithm. *The Visual Computer*, 20(1), 4–16. 32

[Cohen-Steiner et al., 2002] Cohen-Steiner, D., De Verdiere, E., & Yvinec, M. (2002). Conforming delaunay triangulations in 3d. In *Proceedings of the eighteenth annual symposium on Computational geometry* (pp. 199–208).: ACM. 51

[Collins, 1996] Collins, R. (1996). A space-sweep approach to true multi-image matching. In *1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96* (pp. 358–363). 23

[Curless, 1997] Curless, B. (1997). *New methods for surface reconstruction from range images*. PhD thesis, Citeseer. 16

[Curless & Seitz, 2000] Curless, B. & Seitz, S. (2000). 3d photography. *Course Notes for SIGGRAPH 2000*. 15

[Daniels et al., 2007] Daniels, J. I., Ha, L. K., Ochotta, T., & Silva, C. T. (2007). Robust smooth feature extraction from point clouds. In *Proc. of SMI '07* (pp. 123–136). 78, 79, 85, 105

[de Berg et al., 2008] de Berg, M., Cheong, O., Overmars, M., & Van Kreveld, M. (2008). *Computational Geometry – Algorithms and Aplications (3rd)*. Springer-Verlag Berlin. 7

[Delaunay, 1934] Delaunay, B. (1934). Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7, 793–800. 8, 50

[Demarsin et al., 2007] Demarsin, K., Vanderstraeten, D., Volodine, T., & Roose, D. (2007). Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Comput. Aided Des.*, 39(4), 276–283. 78

[Dervieux & Thomasset, 1980] Dervieux, A. & Thomasset, F. (1980). A finite element method for the simulation of a rayleigh-taylor instability. *Approximation methods for Navier-Stokes problems*, (pp. 145–158). 44

[Dey et al., 2005] Dey, T., Giesen, J., Ramos, E., & Sadri, B. (2005). Critical points of the distance to an epsilon-sampling of a surface and flow-complex-based surface reconstruction. In *Proceedings of the twenty-first annual symposium on Computational geometry* (pp. 227).: ACM. 31

[Dey & Sun, 2005] Dey, T. & Sun, J. (2005). An adaptive mls surface for reconstruction with guarantees. In *Proceedings of the third Eurographics symposium on Geometry processing* (pp.43).: Eurographics Association. 40

[Dey, 2004] Dey, T. K. (2004). Curve and surface reconstruction. 29

[Dey & Giesen, 2001] Dey, T. K. & Giesen, J. (2001). Detecting undersampling in surface reconstruction. In *Proceedings of the seventeenth annual symposium on Computational geometry* (pp. 257–263).: ACM. 34

[Dey et al., 2001a] Dey, T. K., Giesen, J., Goswami, S., Hudson, J., Wenger, R., & Zhao, W. (2001a). Undersampling and oversampling in sample based shape modeling. In *Visualization, 2001. VIS'01. Proceedings* (pp. 83–545). 34

[Dey et al., 2003] Dey, T. K., Giesen, J., Goswami, S., & Zhao, W. (2003). Shape dimension and approximation from samples. *Discrete and Computational Geometry*, 29(3), 419–434. 12

[Dey et al., 2001b] Dey, T. K., Giesen, J., & Hudson, J. (2001b). Decimating samples for mesh simplification. In *Proc. 13th Canadian conference on computational geometry* (pp. 85–88).: Citeseer. 34

[Dey et al., 2001c] Dey, T. K., Giesen, J., & Hudson, J. (2001c). Delaunay based shape reconstruction from large data. In *IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics, 2001. Proceedings* (pp. 19–146). 34

[Dey & Goswami, 2003] Dey, T. K. & Goswami, S. (2003). Tight cocone: a water-tight surface reconstructor. *Journal of Computing and Information Science in Engineering*, 3, 302. 34

[Dey & Goswami, 2006] Dey, T. K. & Goswami, S. (2006). Provable surface reconstruction from noisy samples. *Computational Geometry*, 35(1-2), 124–141. 34, 35

[Dinh et al., 2001] Dinh, H., Turk, G., & Slabaugh, G. (2001). Reconstructing surfaces using anisotropic basis functions. In *Proc. of ICCV '01* (pp. 606–613). 78

[Du & Wang, 2006] Du, Q. & Wang, D. (2006). Recent progress in robust and quality delaunay mesh generation. *Journal of Computational and Applied Mathematics*, 195(1-2), 8–23. 47, 49

[Edelsbrunner, 1992] Edelsbrunner, H. (1992). Weighted alpha shapes. 32

[Edelsbrunner, 2003] Edelsbrunner, H. (2003). Surface reconstruction by wrapping finite sets in space. *Discrete and Computational GeometryThe Goodman-Pollack Festschrift*, (pp. 379–404). 31

[Edelsbrunner et al., 1983] Edelsbrunner, H., Kirkpatrick, D., & Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4), 551–559. 30

[Edelsbrunner & Mücke, 1994] Edelsbrunner, H. & Mücke, E. (1994). Three-Dimensional Alpha Shapes. *ACM Transactions on Graphics*, 13(1), 43–72. 31, 32

[Edelsbrunner & Shah, 1997] Edelsbrunner, H. & Shah, N. (1997). Triangulating topological spaces. *International Journal of Computational Geometry & Applications*, 7(4), 365–378. 13, 30

[Everett, 1995] Everett, H. R. (1995). *Sensors for mobile robots: theory and application*. Natick, MA, USA: A. K. Peters, Ltd. 16

[Faugeras & Keriven, 1998a] Faugeras, O. & Keriven, R. (1998a). Complete dense stereovision using level set methods. *Computer VisionECCV'98*, (pp. 379). 56

[Faugeras & Keriven, 1998b] Faugeras, O. & Keriven, R. (1998b). Variational principles, surface evolution, pdes, level set methods, and the stereo problem. *IEEE Transactions on Image Processing*, 7(3). 56

[Faugeras et al., 2001] Faugeras, O., Luong, Q., & Papadopoulou, T. (2001). The geometry of multiple images: The laws that govern the formation of images of a scene and some of their applications. 18

[Fleishman et al., 2005] Fleishman, S., Cohen-Or, D., & Silva, C. (2005). Robust moving least-squares fitting with sharp features. In *Proc. of ACM SIGGRAPH '05* (pp. 552). 40, 79

[Frahm et al., 2010] Frahm, J., Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y., Dunn, E., Clipp, B., Lazebnik, S., et al. (2010). Dense Reconstruction from Millions of Images on a Single PC. 24

[Franke & Nielson, 1980] Franke, R. & Nielson, G. (1980). Smooth interpolation of large sets of scattered data. *International Journal for Numerical Methods in Engineering*, 15(11), 1691–1704. 40

[Frey et al., 1996] Frey, P., Borouchaki, H., & George, P. (1996). Delaunay tetrahedralization using an advancing-front approach. *5th International Meshing Roundtable, SAND 95-2130, Sandia National Laboratories*, (pp. 31–46). 47

[Frey et al., 1998] Frey, P., Borouchaki, H., & George, P. (1998). 3d delaunay mesh generation coupled with an advancing-front approach. *Computer methods in applied mechanics and engineering*, 157(1-2), 115–131. 47

[Frey & George, 2008] Frey, P. & George, P. (2008). *Mesh generation: application to finite elements (Second Edition)*. ISTE Ltd and John Wiley & Sons, Inc. 47

[Frey, 1987] Frey, W. (1987). Selective refinement: a new strategy for automatic node placement in graded triangular meshes. *International Journal for Numerical Methods in Engineering*, 24(11), 2183–2200. 50

[Funke & Ramos, 2002] Funke, S. & Ramos, E. (2002). Smooth-surface reconstruction in near-linear time. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 790).: Society for Industrial and Applied Mathematics. 12

[Furukawa et al., 2010] Furukawa, Y., Curless, B., Seitz, S., Szeliski, R., & Szeliski, R. (2010). Towards internet-scale multi-view stereo. *Image*, 1, I2. 24

[Furukawa & Ponce, 2007] Furukawa, Y. & Ponce, J. (2007). Accurate, dense, and robust multiview stereopsis. In *Proc. of CVPR '07*. 45, 57, 71

[Gargallo & Sturm, 2005] Gargallo, P. & Sturm, P. (2005). Bayesian 3d modeling from images using multiple depth maps. 56

[George, 1971] George, J. (1971). Computer implementation of the finite element method. 49

[George & Seveno, 1994] George, P. & Seveno, E. (1994). The advancing-front mesh generation method revisited. *International Journal for Numerical Methods in Engineering*, 37(21), 3605–3619. 49

[Giesen & John, 2003] Giesen, J. & John, M. (2003). The flow complex: a data structure for geometric modeling. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 285–294).: Society for Industrial and Applied Mathematics. 31

[Goesele et al., 2007] Goesele, M., Snavely, N., Curless, B., Hoppe, H., & Seitz, S. (2007). Multi-view stereo for community photo collections. In *Proc. of ICCV '07*. 24, 45, 56

[Gopi et al., 2000] Gopi, M., Krishnan, S., & Silva, C. (2000). Surface reconstruction based on lower dimensional localized delaunay triangulation. In *Computer Graphics Forum*, volume 19 (pp. 467–478).: John Wiley & Sons. 33

[Grenander & Miller, 1994] Grenander, U. & Miller, M. (1994). Representations of knowledge in complex systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 56(4), 549–603. 57

[Guennebaud & Gross, 2007] Guennebaud, G. & Gross, M. (2007). Algebraic point set surfaces. In *Proc. of ACM SIGGRAPH '07* (pp.2̃3). 78, 79, 88

[Gumhold et al., 2001] Gumhold, S., Wang, X., & MacLeod, R. (2001). Feature extraction from point clouds. In *Proceedings of the 10th international meshing roundtable*, volume 2001 (pp. 293–305). 78

[Guy & Medioni, 1997] Guy, G. & Medioni, G. (1997). Inference of surfaces, 3 d curves, and junctions from sparse, noisy, 3 d data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11), 1265–1277. 78

[Harris & Stephens, 1988] Harris, C. & Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15 (pp.5̃0).: Manchester, UK. 21

[Hartley & Zisserman, 2004] Hartley, R. & Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. New York, NY, USA: Cambridge University Press. 18, 23

[Hermeline, 1982] Hermeline, F. (1982). Triangulation automatique d'un poly edre en dimension n. *RAIRO numerical analysis*, 16(3). 50

[Hilton, 2005] Hilton, A. (2005). Scene modelling from sparse 3d data. *Image and Vision Computing*, 23(10), 900 – 920. 57

[Hoppe et al., 1992] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., & Stuetzle, W. (1992). Surface reconstruction from unorganized points. *Proc. of ACM SIGGRAPH '92*, (pp. 71–71). 37, 93, 116

[Hornung & Kobbelt, 2006] Hornung, A. & Kobbelt, L. (2006). Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information. In *Proc. of SGP '06* (pp. 41–50). 43, 55

[Huang et al., 2009] Huang, H., Li, D., Zhang, H., Ascher, U., & Cohen-Or, D. (2009). Consolidation of unorganized point clouds for surface reconstruction. *ACM Transactions on Graphics (TOG)*, 28(5), 1–7. 45

[Jachiet et al., 2010] Jachiet, A.-L., Labatut, P., & Pons, J.-P. (2010). Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *Conference on Computer Vision and Pattern Recognition (CVPR)* San Francisco. 57

[Jancosek et al., 2009] Jancosek, M., Shekhovtsov, A., & Pajdla, T. (2009). Scalable multi-view stereo. *3DIM 2009 (ICCV workshop)*. 24

[Jenke et al., 2008] Jenke, P., Wand, M., Straßer, W., & AKA, A. (2008). Patch-graph reconstruction for piecewise smooth surfaces. *Proc. of VMV '08*, (pp.3̃). 79, 105

[Ju et al., 2002] Ju, T., Losasso, F., Schaefer, S., & Warren, J. (2002). Dual contouring of hermite data. *ACM Transactions on Graphics (TOG)*, 21(3), 346. 49

[Kass et al., 1988] Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: Active contour models. *International journal of computer vision*, 1(4), 321–331. 44

[Kazhdan, 2005] Kazhdan, M. (2005). Reconstruction of solid models from oriented point sets. In *Proceedings of the third Eurographics symposium on Geometry processing* (pp.7̃3).: Eurographics Association. 41, 42

[Kazhdan et al., 2006] Kazhdan, M., Bolitho, M., & Hoppe, H. (2006). Poisson surface reconstruction. In *Proc. of SGP '06*. 42, 45, 55, 57, 78, 88, 91, 96

[Kobbelt & Botsch, 2004] Kobbelt, L. & Botsch, M. (2004). A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6), 801–814. 39

[Kobbelt et al., 2001] Kobbelt, L., Botsch, M., Schwanecke, U., & Seidel, H. (2001). Feature sensitive surface extraction from volume data. In *Proc. of ACM SIGGRAPH '01* (pp. 57–66). 36, 48, 52, 79, 97

[Kolluri, 2008] Kolluri, R. (2008). Provably good moving least squares. *ACM Transactions on Algorithms (TALG)*, 4(2), 1–25. 40

[Kolluri et al., 2004] Kolluri, R., Shewchuk, J. R., & O'Brien, J. F. (2004). Spectral surface reconstruction from noisy point clouds. In *Proc. of SGP '04* (pp. 11–21). 35

[Kolmogorov & Zabih, 2002] Kolmogorov, V. & Zabih, R. (2002). Multi-camera scene reconstruction via graph cuts. *Lecture Notes in Computer Science*, (pp. 82–96). 56

[Kutulakos & Seitz, 2000] Kutulakos, K. & Seitz, S. (2000). A theory of shape by space carving. *International Journal of Computer Vision*, 38(3), 199–218. 56

[Labatut, 2009] Labatut, P. (2009). *Labeling of data-driven complexes for surface reconstruction*. PhD thesis, Université Paris Diderot-Paris VII. 15, 22, 25

[Labatut et al., 2007] Labatut, P., Pons, J.-P., & Keriven, R. (2007). Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. *Proc. of ICCV '07*, (pp. 1–8). 21, 45, 57

[Lafarge et al., 2010] Lafarge, F., Keriven, R., Brdif, M., & Vu, H.-H. (2010). Hybrid multi-view reconstruction by jump-diffusion. In *Conference on Computer Vision and Pattern Recognition (CVPR)* San Fransisco. 57

[Lancaster & Salkauskas, 1981] Lancaster, P. & Salkauskas, K. (1981). Surfaces generated by moving least squares methods. *Mathematics of computation*, 37(155), 141–158. 39

[Lau & Lo, 1996] Lau, T. & Lo, S. (1996). Finite element mesh generation over analytical curved surfaces. *Computers & structures*, 59(2), 301–309. 49

[Lee, 1999] Lee, C. (1999). Automatic adaptive mesh generation using metric advancing front approach. *Engineering Computations*, 16(2), 230–265. 49

[Lee, 2000] Lee, I. (2000). Curve reconstruction from unorganized points. *Computer Aided Geometric Design*, 17(2), 161–177. 85

[Lempitsky & Boykov, 2007] Lempitsky, V. & Boykov, Y. (2007). Global optimization for shape fitting. In *2007 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8).: IEEE. 43

[Lempitsky et al., 2006] Lempitsky, V., Boykov, Y., & Ivanov, D. (2006). Oriented visibility for multiview reconstruction. *Computer Vision–ECCV 2006*, (pp. 226–238). 56

[Levin, 1998] Levin, D. (1998). The approximation power of moving least-squares. *Mathematics of computation*, 67(224), 1517–1531. 39

[Levin, 2003] Levin, D. (2003). Mesh-independent surface interpolation. *Geometric Modeling for Scientific Visualization*, 3, 37–49. 39

[Lévy & Liu, 2010] Lévy, B. & Liu, Y. (2010). Lp centroidal voronoi tesselation and its applications. *ACM Transactions on Graphics (SIGGRAPH conference proceedings)*. 51, 80

[Lindeberg, 1993] Lindeberg, T. (1993). *Scale-space theory in computer vision*. Springer. 20

[Lindeberg & Gårding, 1997] Lindeberg, T. & Gårding, J. (1997). Shape-adapted smoothing in estimation of 3-d shape cues from affine deformations of local 2-d brightness structure* 1. *Image and Vision Computing*, 15(6), 415–434. 20

[Lipman et al., 2007] Lipman, Y., Cohen-Or, D., & Levin, D. (2007). Data-dependent mls for faithful surface approximation. In *Proc. of SGP '07* (pp.67̃). 79

[Lo, 1985] Lo, S. (1985). A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering*, 21(8), 1403–1426. 49

[Lo, 1991] Lo, S. (1991). Volume discretization into tetrahedra–ii. 3d triangulation by advancing front approach. *Computers & Structures*, 39(5), 501–511. 49

[Lo & Lau, 1998] Lo, S. & Lau, T. (1998). Mesh generation over curved surfaces with explicit control on discretization error. *Engineering Computations*, 15(2), 357–373. 49

[Löhner, 1996] Löhner, R. (1996). Progress in grid generation via the advancing front technique. *Engineering with Computers*, 12(3), 186–210. 49

[Löhner & Parikh, 1988] Löhner, R. & Parikh, P. (1988). Generation of three-dimensional unstructured grids by the advancing-front method. *International Journal for Numerical Methods in Fluids*, 8(10), 1135–1149. 49

[Lorensen & Cline, 1987] Lorensen, W. & Cline, H. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (pp. 169).: ACM. 36, 48

[Lourakis & Argyros, 2009] Lourakis, M. A. & Argyros, A. (2009). SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1), 1–30. 18

[Lowe, 2004] Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 91–110. 21

[Luong & Faugeras, 1996] Luong, Q. & Faugeras, O. (1996). The fundamental matrix: Theory, algorithms, and stability analysis. *International Journal of Computer Vision*, 17(1), 43–75. 19

[Manessis et al., 2000] Manessis, A., Hilton, A., Palmer, P., McLauchlan, P., & Shen, X. (2000). Reconstruction of scene models from sparse 3d structure. *Proc. of CVPR '00*, 2, 2666. 57

[Marcum & Weatherill, 1995] Marcum, D. & Weatherill, N. (1995). Unstructured grid generation using iterative point insertion and local reconnection. *AIAA journal*, 33(9), 1619–1625. 47

[Marr & Hildreth, 1980] Marr, D. & Hildreth, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 207(1167), 187–217. 21

[Mavriplis, 1995] Mavriplis, D. (1995). An Advancing Front Delaunay Triangulation Algorithm Designed for Robustness. *Journal of Computational Physics*, 117(1), 90–101. 47

[Mederos et al., 2005] Mederos, B., Amenta, N., Velho, L., & de Figueiredo, L. (2005). Surface reconstruction from noisy point clouds. In *Proceedings of the third Eurographics symposium on Geometry processing* (pp.53).: Eurographics Association. 35

[Mederos et al., 2003] Mederos, B., Velho, L., & De Figueiredo, L. (2003). Moving least squares multiresolution surface approximation. In *XVI Brazilian Symposium on Computer Graphics and Image Processing, 2003. SIBGRAPI 2003* (pp. 19–26). 40

[Merigot et al., 2009] Merigot, Q., Ovsjanikov, M., & Guibas, L. (2009). Robust voronoi-based curvature and feature estimation. In *Proc. of SIAM/ACM SPM '09* (pp. 1–12). 79, 80, 82, 83, 85, 91, 93, 105

[Merlo et al., 2002] Merlo, S., Norgia, M., & Donati, S. (2002). Fiber gyroscope principles. *Handbook of optical fibre sensing technology*, (pp. 331). 26

[Merrell et al., 2007] Merrell, P., Akbarzadeh, A., Wang, L., Mordohai, P., Frahm, J., Yang, R., Nistér, D., & Pollefeys, M. (2007). Real-time visibility-based fusion of depth maps. In *Proceedings of International Conf. on Computer Vision*: Citeseer. 56

[Merriam, 1991] Merriam, M. (1991). An efficient advancing front algorithm for delaunay triangulation. In *AIAA, Aerospace Sciences Meeting*. 47

[Mitchell & Vavasis, 1992] Mitchell, S. & Vavasis, S. (1992). Quality mesh generation in three dimensions. In *Proceedings of the eighth annual symposium on Computational geometry* (pp. 221).: ACM. 48

[Mitchell & Vavasis, 2000] Mitchell, S. & Vavasis, S. (2000). Quality mesh generation in higher dimensions. *SIAM Journal on Computing*, 29(4), 1334–1370. 48

[Mitra et al., 2004] Mitra, N., Nguyen, A., & Guibas, L. (2004). Estimating surface normals in noisy point cloud data. *International Journal of Computational Geometry and Applications*, 14(4), 261–276. 45

[Morris & Kanade, 2000] Morris, D. & Kanade, T. (2000). Image-consistent surface triangulation. In *Proc. of CVPR '00*. 56, 57

[Morse et al., 2005] Morse, B., Yoo, T., Rheingans, P., Chen, D., & Subramanian, K. (2005). Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *ACM SIGGRAPH 2005 Courses* (pp.78).: ACM. 41

[Mount & Arya, 2010] Mount, D. M. & Arya, S. (2010). ANN, a library for Approximate Nearest Neighbor searching. http://www.cs.umd.edu/~mount/ANN/. 93

[Mullen et al., 2010] Mullen, P., De Goes, F., Desbrun, M., Cohen-Steiner, D., & Alliez, P. (2010). Signing the Unsigned: Robust Surface Reconstruction from Raw Pointsets. *Computer Graphics Forum*, 29, 1733–1741. 38, 55, 98

[Nagai et al., 2009] Nagai, Y., Ohtake, Y., & Suzuki, H. (2009). Smoothing of partition of unity implicit surfaces for noise robust surface reconstruction. In *SGP '09: Proceedings of the Symposium on Geometry Processing* (pp. 1339–1348). Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. 39

[Nakatuji et al., 2005] Nakatuji, A., Sugaya, Y., & Kanatani, K. (2005). Mesh optimization using an inconsistency detection template. *Computer Vision, IEEE International Conference on*, 2, 1148–1153. 56

[Needleman & Wunsch, 1970] Needleman, S. B. & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48. 65

[Nistér, 2004] Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6), 756–777. 18

[Ohtake et al., 2005a] Ohtake, Y., Belyaev, A., & Alexa, M. (2005a). Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. In *Proc. of SGP '05* (pp. 149). 36, 78

[Ohtake et al., 2003a] Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., & Seidel, H.-P. (2003a). Multi-level partition of unity implicits. In *Proc. of ACM SIGGRAPH*. 38, 55, 79

[Ohtake et al., 2003b] Ohtake, Y., Belyaev, A., & Seidel, H. (2003b). A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions. 41

[Ohtake et al., 2004] Ohtake, Y., Belyaev, A., & Seidel, H. (2004). 3d scattered data approximation with adaptive compactly supported radial basis functions. In *Shape Modeling Applications, 2004. Proceedings* (pp. 31–39). 41

[Ohtake et al., 2005b] Ohtake, Y., Belyaev, A., & Seidel, H.-P. (2005b). An integrating approach to meshing scattered point data. In *ACM Symposium on Solid and Physical Modeling*. 32

[Osher & Sethian, 1988] Osher, S. & Sethian, J. (1988). Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1), 12–49. 44

[Owen, 1998] Owen, S. (1998). A survey of unstructured mesh generation technology. In *7th International Meshing Roundtable*, volume 3: Citeseer. 47

[Oztireli et al., 2009] Oztireli, C., Guennebaud, G., & Gross, M. (2009). Feature preserving point set surfaces based on non-linear kernel regression. In *Computer Graphics Forum*, volume 28 (pp. 493–501). 78

[Pauly et al., 2003] Pauly, M., Keiser, R., & Gross, M. (2003). Multi-scale feature extraction on point-sampled surfaces. In *Computer Graphics Forum*, volume 22 (pp. 281–289). 78

[Phillips et al., 1993] Phillips, M. et al. (1993). Geomview manual. *The Geometry Center*. 109

[Pollefeys et al., 2008] Pollefeys, M., Nister, D., Frahm, J., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S., Merrell, P., et al. (2008). Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*. 24, 56

[Pollefeys et al., 2004] Pollefeys, M., Van Gool, L., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., & Koch, R. (2004). Visual modeling with a hand-held camera. *Int. J. Comput. Vision*, 59(3), 207–232. 18, 56

[Pons et al., 2007] Pons, J.-P., Keriven, R., & Faugeras, O. (2007). Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *Int. J. Comput. Vision*, 72(2), 179–193. 56

[Rassineux, 1998] Rassineux, A. (1998). Generation and optimization of tetrahedral meshes by advancing front technique. *International Journal for Numerical Methods in Engineering*, 41(4), 651–674. 49

[Reuter et al., 2005] Reuter, P., Joyot, P., Trunzler, J., Boubekeur, T., Schlick, C., LIPSI, E., & Bidart, F. (2005). Surface reconstruction with enriched reproducing kernel particle approximation. In *Proc. Symposium Point-Based Graphics, 2005* (pp. 79–87). 79

[Rineau & Yvinec, 2010] Rineau, L. & Yvinec, M. (2010). 3D surface mesh generation. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.6 edition. http //www.cgal.org/Manual/3.6/doc_html/cgal_manual/packages.html#Pkg SurfaceMesher3. 69

[Ruppert, 1993] Ruppert, J. (1993). A new and simple algorithm for quality 2-dimensional mesh generation. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms* (pp. 83–92).: Society for Industrial and Applied Mathematics. 51

[Ruppert, 1995] Ruppert, J. (1995). A delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of algorithms*, 18(3), 548–585. 51

[Salman & Yvinec, 2009] Salman, N. & Yvinec, M. (2009). High resolution surface reconstruction from overlapping multiple-views. In *SCG '09: Proceedings of the 25th annual symposium on Computational geometry* (pp. 104–105). New York, NY, USA: ACM. 76

[Salman & Yvinec, 2009] Salman, N. & Yvinec, M. (2009). Surface Reconstruction from Multi-View Stereo. *Lecture notes in computer science*. 76

[Salman & Yvinec, 2010] Salman, N. & Yvinec, M. (2010). Surface Reconstruction from Multi-View Stereo of Large-Scale Outdoor Scenes. *International Journal of Virtual Reality*, Volume 9, 19–26. Special issue for ACCV09. 76

[Salman et al., 2010] Salman, N., Yvinec, M., & Mérigot, Q. (2010). Feature Preserving Mesh Generation from 3D Point Clouds. In Olga Sorkine & Bruno Levy (Eds.), *Symposium on Geometry Processing* Lyon France. 52, 99

[Samozino et al., 2006] Samozino, M., Alexa, M., Alliez, P., & Yvinec, M. (2006). Reconstruction with voronoi centered radial basis functions. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing* (pp. 51–60). Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. 41

[Savchenko et al., 1995] Savchenko, V., Pasko, A., Okunev, O., & Kunii, T. (1995). Function representation of solids reconstructed from scattered surface points and contours. In *Computer Graphics Forum*, volume 14 (pp. 181–188).: John Wiley & Sons. 41

[Schall et al., 2006] Schall, O., Belyaev, A., & Seidel, H. (2006). Adaptive fourier-based surface reconstruction. *Geometric Modeling and Processing-GMP 2006*, (pp. 34–44). 42

[Schall et al., 2007] Schall, O., Belyaev, A., & Seidel, H. (2007). Error-guided adaptive fourier-based surface reconstruction. *Computer-Aided Design*, 39(5), 421–426. 42

[Scheidegger et al., 2005] Scheidegger, C., Fleishman, S., & Silva, C. (2005). Triangulating point set surfaces with bounded error. In *Proceedings of the third Eurographics symposium on Geometry processing* (pp.6̃3).: Eurographics Association. 40, 49

[Schreiner et al., 2006] Schreiner, J., Scheidegger, C., Fleishman, S., & Silva, C. (2006). Direct (re) meshing for efficient surface processing. In *Computer Graphics Forum*, volume 25 (pp. 527–536). 49, 52, 79

[Schroeder & Shephard, 1990] Schroeder, W. & Shephard, M. (1990). A combined octree/delaunay method for fully automatic 3-d mesh generation. *International Journal for Numerical Methods in Engineering*, 29(1), 37–55. 47

[Seitz & Dyer, 2002] Seitz, S. & Dyer, C. (2002). Photorealistic scene reconstruction by voxel coloring. US Patent 6,363,170. 56

[Seitz et al., 2006] Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., & Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. of CVPR '06*. 1, 16, 56, 57, 105

[Sharf et al., 2006] Sharf, A., Lewiner, T., Shamir, A., Kobbelt, L., & Cohen Or, D. (2006). Competing fronts for coarse-to-fine surface reconstruction. In *Computer Graphics Forum*, volume 25 (pp. 389–398).: Citeseer. 44

[Shen et al., 2004] Shen, C., O'Brien, J. F., & Shewchuk, J. R. (2004). Interpolating and approximating implicit surfaces from polygon soup. In *Proc. of ACM SIGGRAPH '04* (pp. 204). 39

[Shephard & Georges, 1991] Shephard, M. & Georges, M. (1991). Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering*, 32(4), 709–749. 48

[Shewchuk, 1996] Shewchuk, J. (1996). Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. *Applied Computational Geometry Towards Geometric Engineering*, (pp. 203–222). 51

[Shewchuk, 1998] Shewchuk, J. (1998). Tetrahedral mesh generation by delaunay refinement. In *Proceedings of the fourteenth annual symposium on Computational geometry* (pp. 86–95).: ACM. 51

[Shewchuk, 2000] Shewchuk, J. (2000). Mesh generation for domains with small angles. In *Proc. of SCG '00* (pp.1̃0). 51, 80

[Shewchuk, 2002] Shewchuk, J. (2002). Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry*, 22(1-3), 21–74. 51

[Shewchuk et al., 1997] Shewchuk, J., Miller, G., & David, R. (1997). Delaunay refinement mesh generation. 47

[Strecha et al., 2004] Strecha, C., Fransens, R., & Van Gool, L. (2004). Wide-baseline stereo from multiple views: a probabilistic account. In *Proc. of CVPR '04*. 56, 71

[Strecha et al., 2006] Strecha, C., Fransens, R., & Van Gool, L. (2006). Combined depth and outlier estimation in multi-view stereo. In *Proc. of CVPR '06*. 56, 71

[Strecha et al., 2008] Strecha, C., von Hansen, W., Van Gool, L., Fua, P., & Thoennessen, U. (2008). On benchmarking camera calibration and multi-view stereo for high resolution imagery. *Proc. of CVPR '08*. http://cvlab.epfl.ch/ strecha/multiview/denseMVS.html. 16, 25, 55, 57, 63, 65, 69, 71, 72, 73, 75, 105, 106

[Tobor et al., 2004] Tobor, I., Reuter, P., & Schlick, C. (2004). Efficient reconstruction of large scattered geometric datasets using the partition of unity and radial basis functions. *Journal of WSCG 2004*, 12(3), 467–474. 39, 41

[Toldo & Fusiello, 2010] Toldo, R. & Fusiello, A. (2010). Photo-consistent planar patches from unstructured cloud of points. 74, 104

[Tournois et al., 2009] Tournois, J., Wormser, C., Alliez, P., & Desbrun, M. (2009). Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Transactions on Graphics (TOG)*, 28(3), 75. 51

[Tran & Davis, 2006] Tran, S. & Davis, L. (2006). 3d surface reconstruction using graph cuts with surface constraints. *Computer Vision–ECCV 2006*, (pp. 219–231). 56

[Triggs et al., 2000] Triggs, B., McLauchlan, P., Hartley, R., & Fitzgibbon, A. (2000). Bundle adjustmenta modern synthesis. *Vision algorithms: theory and practice*, (pp. 153–177). 18

[Turk & O'Brien, 1999] Turk, G. & O'Brien, J. (1999). Variational implicit surfaces. *Techical Reports GIT-GVU-99-15*. 41

[Turk & O'brien, 2002] Turk, G. & O'brien, J. (2002). Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics (TOG)*, 21(4), 873. 41

[Tylecek & Sara, 2009] Tylecek, R. & Sara, R. (2009). Depth map fusion with camera position refinement. In *Computer Vision Winter Workshop*. 71

[Veltkamp, 1995] Veltkamp, R. (1995). Boundaries through scattered points of unknown density. *Graphical Models and Image Processing*, 57(6), 441–452. 30

[Vogiatzis et al., 2007] Vogiatzis, G., Esteban, C., Torr, P., & Cipolla, R. (2007). Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE transactions on pattern analysis and machine intelligence*, (pp. 2241–2246). 56

[Vogiatzis et al., 2005] Vogiatzis, G., Torr, P., & Cipolla, R. (2005). Multi-view stereo via volumetric graph-cuts. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005* (pp. 391–398). 56

[Vu et al., 2009] Vu, H., Keriven, R., Labatut, P., & Pons, J.-P. (2009). Towards high-resolution large-scale multi-view stereo. In *Proc. of CVPR '09*. 20, 21, 23, 45, 57, 59, 60, 71, 74, 104

[Walder et al., 2005] Walder, C., Chapelle, O., & Schölkopf, B. (2005). Implicit surface modelling as an eigenvalue problem. In *Proceedings of the 22nd international conference on Machine learning* (pp. 939).: ACM. 45

[Watson, 1981] Watson, D. (1981). Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The computer journal*, 24(2), 167. 50

[Weber et al., 2010] Weber, C., Hahmann, S., & Hagen, H. (2010). Sharp feature detection in point clouds. In *IEEE International Conference on Shape Modeling and Applications, 2010. SMI'10*. 79

[Wu & Kobbelt, 2004] Wu, J. & Kobbelt, L. (2004). Optimized sub-sampling of point sets for surface splatting. In *Computer Graphics Forum*, volume 23 (pp. 643–652).: John Wiley & Sons. 18

[Xie et al., 2003] Xie, H., Wang, J., Hua, J., Qin, H., & Kaufman, A. (2003). Piecewise c1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (pp.13).: IEEE Computer Society. 40

[Yan et al., 2009] Yan, D., Lévy, B., Liu, Y., Sun, F., & Wang, W. (2009). Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. In *Proceedings of the Symposium on Geometry Processing* (pp. 1445–1454).: Eurographics Association. 51

[Yang et al., 2002] Yang, R., Welch, G., & Bishop, G. (2002). Real-time consensus-based scene reconstruction using commodity graphics hardware. 23

[Yerry & Shephard, 1983] Yerry, M. & Shephard, M. (1983). A modified quadtree approach to finite element mesh generation. *IEEE Computer Graphics and Applications*, 3(1), 39–46. 48

[Yerry & Shephard, 1984] Yerry, M. & Shephard, M. (1984). Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering*, 20(11), 1965–1990. 48

[Zach et al., 2007] Zach, C., Pock, T., & Bischof, H. (2007). A globally optimal algorithm for robust tv-l 1 range image integration. In *Proc. of ICCV '07*, volume 1. 56

[Zaharescu et al., 2007] Zaharescu, A., Boyer, E., & Horaud, R. (2007). Transformesh: a topology-adaptive mesh-based approach to surface evolution. In *Proc. of ACCV '07*. 71

[Zhao et al., 2001] Zhao, H., Osher, S., & Fedkiw, R. (2001). Fast surface reconstruction using the level set method. In *IEEE Workshop on Variational and Level Set Methods in Computer Vision, 2001. Proceedings* (pp. 194–201). 31, 44