Inverse geometry : from the raw point cloud to the 3d surface : theory and algorithms

Julie Digne

École Normale Supérieure de Cachan

Thèse

Présentée par

Julie DIGNE

pour l'obtention du titre de

Docteur de l'École Normale Supérieure de Cachan
Spécialité : MATHÉMATIQUES APPLIQUÉES

Géométrie inverse: du nuage de points brut à la surface 3D Théorie et Algorithmes

Inverse Geometry: from the Raw Point Cloud to the 3D Surface Theory and Algorithms

Directeur de thèse: JEAN-MICHEL MOREL
soutenue le 23 Novembre 2010 à l'ENS Cachan

Jury :

Président :	Yves MEYER	-	CMLA - ENS Cachan
Rapporteurs :	Pierre ALLIEZ	-	INRIA Sophia Antipolis
	Ron KIMMEL	-	Technion
	Guillermo SAPIRO	-	University of Minnesota
Directeur :	Jean-Michel MOREL	-	CMLA - ENS Cachan
Examinateurs :	Tamy BOUBEKEUR	-	Telecom ParisTech
	Frédéric CHAZAL	-	INRIA Saclay
	Claire LARTIGUE	-	LURPA - ENS Cachan
Invité :	Jacques BLANC-TALON	-	DGA

Remerciements

Trois ans de travail sont résumés dans les pages qui vont suivre. Cela semble presque trop peu (les lecteurs me contrediront sans doute sur ce point).

Je voudrais remercier Jean-Michel Morel de m'avoir dirigée pendant ces trois ans. Cette thèse a été riche en apprentissages grâce à sa grande disponibilité et sa patience. J'aimerais remercier mon jury : Pierre Alliez, Ron Kimmel et Guillermo Sapiro qui ont accepté de rapporter ma thèse, Jacques Blanc-Talon, Tamy Boubekeur, Frédéric Chazal, Claire Lartigue et Yves Meyer qui ont accepté de faire partie du jury. Cette thèse a été elaborée autour des données fournies par le LURPA que je remercie vivement. J'aimerais aussi remercier la DGA pour avoir financé ces trois ans de recherche.

Merci à tous les chercheurs rencontrés au cours de cette thèse et avec qui j'ai pu avoir des conversations très intéressantes : Freddy Bruckstein, Guillermo Sapiro, Pierre Alliez, Frédéric Chazal, Steve Oudot, José-Luis Lisani, Tony Buades, Pascal Monasse et toute l'équipe du projet MISS.

Travailler au CMLA a été un vrai plaisir, merci en particulier à Véronique Almadovar, Micheline Brunetti, Sandra Doucet, Virginie Pauchont et Carine Saint-Prix pour leur efficacité et la bonne humeur qu'elles répandent dans le laboratoire.

Tous mes remerciements vont aux thésards qui m'ont accompagnée pendant ces trois ans : Neus Sabater pour des séances piscines intensives et des conseils cuisine pour couronner le tout. Eric Bughin, pour la pause café du matin. Aude Champmartin, pour me rappeler que, non, au labo tout le monde n'est pas sous linux et me trainer à la gym. Adina Ciomaga pour être un peu strrrressée avant les rendez-vous. Bruno Galerne, pour des discussions cinématographiques intenses et Zhongwei pour m'avoir fait comprendre la stéréorectification (enfin!). Merci aussi à Nicolas, parce que la 3D sans problème informatique ne serait plus de la 3D. Merci à Rafa pour une gentillesse et une patience sans borne même quand je repose dix fois les mêmes questions. Merci à tous les thésards et post-docs qui ont rendu la vie au labo très agréable entre pauses café au pavillon des jardins et repas gastronomiques le soir : Adina, Aude, Ayman, Benjamin, Bruno, Eric, Frédéric, Frédérique, Gaël, Ives, Jean-Pascal, Julien, Marc, Mauricio, Miguel, Morgan, Neus, Nicolas C., Nicolas L., Romain, Saad, Yohann et Zhongwei. Merci enfin aux visiteurs : Alex, Anne, Pascal, Gloria et Pablo...

Cette thèse a nécessité :

- La lecture de 398 articles;
- La rédaction de 30 rapports;

- 3 réinstallations de Linux;
- 1 changement de carte mère;
- 1 remplacement de disque dur;
- x plantages généralisés des serveurs de calcul du labo;
- 255009 lignes de code;
- 26 programmes;
- 11 cahiers;
- 25 stylos et crayons divers;
- 1350 litres de thé;
- 126 litres de café;
- 421200 minutes de musique écoutée (dont une bonne moitié de Händel et Rossini).

Je voudrais remercier certains professeurs qui m'ont beaucoup marquée par leur grande pédagogie : Jean-Louis Garcin, Jean-Aristide Cavaillès au lycée Chaptal et Francis Schmitt, disparu trop tôt, à Télécom ParisTech. Mes remerciements à Yann Gousseau et Henri Maitre pour leurs conseils d'orientation.

Finalement, je voudrais remercier mes parents et mes soeurs, Jeanne et Charlotte, pour leur soutien sans lequel cette thèse n'aurait pas existé. Merci à Nicolas, qui a réussi à me supporter même pendant que je rédigeais! Merci à Hugo aussi et bon courage pour ta thèse. Merci enfin à tous mes amis télécommiens et non télécommiens, à ma prof de chant, à mon ensemble vocal et à mon club théâtre, parce que la thèse c'est bien, mais l'oublier une ou deux heures ça permet de prendre du recul.

Et comme il ne faut oublier personne, merci à tous ceux qui ne figurent pas dans ces remerciements.

Contents

List of Figures

Notations

- $\mathcal{M}$ denotes a smooth surface
- $\mathcal{M}_S$ denotes a set of samples on $\mathcal{M}$
- p is a point of the smooth surface
- $\mathbf{n}$ is the normal to the surface at p
- $\mathcal{B}_r$ and $\mathcal{C}_r$ are two different types of neighborhoods of p (that will be defined in section 3.2.1)
- r is the radius of the neighborhood
- m is a point of the neighborhood $\mathcal{B}_r$ or $\mathcal{C}_r$
- o is the barycenter of the $\mathcal{B}_r$ or $\mathcal{C}_r$
- k_1 and k_2 are the principal curvatures of the surface at p
- $\mathbf{t}_1$, $\mathbf{t}_2$ are the principal directions of the surface at p
- H is the mean curvature of the surface at p
- V is a mesh and v, v_i vertices of V, T is the set of triangles of V.
- g is a polynomial

Chapter 1

Introduction

Context

This thesis started as a part of a collaboration between two laboratories of the Ecole Normale Supérieure de Cachan, the CMLA (Centre de Mathématiques et Leurs Applications) and the LURPA (Laboratoire Universitaire de Recherche en Productique Automatisée). This project was named *Géométrie Inverse Pour l'Industrie* (Inverse Geometry for the Industry) and aimed at building a closed loop acquisition system. This thesis deals with the data analysis side of this project.

The LURPA high precision scanner is an experimental device delivering very large high precision raw data point sets with up to 35 million points, usually made of about 300 different scan sweeps.

In the course of the project it was progressively realized that the size of the raw point sets, and their precision (nominally about $20\mu m$) raised questions which did not seem to have been addressed before. A first serious problem was that most state of the art techniques were actually too complex to be even tested on the data in reasonable time. Furthermore, the current meshing and analysis techniques were not designed for the conservation and visualization of high precision clouds, and usually lost accuracy and re-sampled or under-sampled the data. Smoothing and re-sampling were acceptable with noisy and only approximately registered clouds obtained by older generation scanners. Yet it seems that new opportunities for the visual exploration and conservation of valuable objects are opening up with future more precise scanners. As usual with better tools the first question is linked to the data proliferation.

This situation led us to a complete rewriting of the processing chain starting with a raw data set consisting of many scans with very irregular sampling and holes. While the end result remains a meshed surface, the steps leading to it have obeyed new requirements aimed at preserving the high accuracy throughout the processing chain:

- to be able to safely orient the raw cloud before meshing (raw orientation);
- to be able to mesh the raw point could itself, thus obtaining a raw mesh (raw meshing);

- to be therefore able to visualize all scanning artifacts in the raw mesh (raw visualization);
- to correct the merging artifacts with minimal resolution loss (merging of raw scans) ;
- to delineate all missing parts (holes) to guide the laser head toward them, or, at the very least, to detect and fill them up automatically;
- to propose a raw cloud scale space analysis and segmentation method, based on the very same process leading to the surface construction.

A last problem arising in the course of this work and worth mentioning has been the scarcity of a shared data base of raw data point sets on which experiment sharing and performance benchmarks could be performed. Most available point clouds are meshed, which usually means that they are already smoothed and re-sampled. Thus most experiments were made on industrial objects and molded copies of archeological objects with fine texture and detail scanned at LURPA. Fortunately, we were also able to test our algorithms on a few raw data point sets provided by Stanford University (*Forma Urbis Romae* Project).

The main contributions of the thesis are reviewed in the next sections. For the sake of clarity, the contributions of the thesis are listed dryly, without the necessary bibliographic analysis. Of course, the bibliographic analysis and experimental comparison will be extensive in the other chapters.

Thesis summary

The problem of surface reconstruction from raw points is considered as essentially solved. Indeed, many commercial devices acquire objects and reconstruct their surface. Nevertheless, the reconstructed surface is usually very smooth as a result of the scanner internal de-noising process and the fusion of different scans. The precision loss is obvious in that all pieces look polished and glossy, having visibly lost their grain and texture. By looking at the academic data sets available on the web, a similar conclusion can be drawn. Surface reconstruction is, in fact, far from being adequately addressed in the context of high precision data.

This thesis has adopted the somewhat extreme conservative position, not to loose nor alter any raw sample throughout the whole processing pipeline. There are two main reasons to try to see directly the raw input samples. First, this could and will serve to control the acquisition loop. The input raw points have to be visualized, to point out all surface imperfections (holes, offsets) and correct them immediately. Second, since high precision data can capture the slightest surface variation, any smoothing, any sub-sampling can loose textural detail. Although

of course further compressing steps may become necessary, building and seeing first the highest quality reference data set before any compression seems to be a sound precaution. It also raises the hope of getting a real 3D microscope, enlarging considerably 3D objects and revealing their details.

The thesis attempts to prove that one can triangulate the raw point cloud with almost no sample loss, and solves the exact visualization problem for large data sets. Although this triangulation process performs remarkably well on accurate data sets, we shall see that it actually can be applied safely on any data set, even on extremely noisy point clouds.

As mentioned, the typical point sets treated here contain around 15 million points (up to 35 million points) and are made of some 300 different scan sweeps. To the best of our knowledge, the bibliographical analysis and numerical comparison will show that no reliable method existed yet for the exact visualization of all the input raw points: most methods either build a mesh by discarding a large part of the input samples, or simply create new vertices and loose the input samples. Large data sets have already been acquired and considered in the literature (Stanford *Digital Michelangelo* and *Fragment Urbis Romae* Project). Yet the available models resulting from these acquisitions are smoothed and re-sampled. They are generally not the initial scans.

To achieve the high precision triangulation which is the main goal of this thesis, two major problems will be solved. The first is the orientation of the complete raw point set, previous to meshing. The second one is the correction of tiny scan misalignments leading, nonetheless, to strong high frequency aliasing and hampering completely a direct visualization. Another outcome is the accurate delineation of scanning holes. The final result aimed at is an accurate visualization of the surface containing all raw points, with low frequencies slightly corrected to avoid aliasing effects, and intact high frequencies.

The second development of the thesis is a general low frequency-high frequency decomposition algorithm for any point cloud. This permits 3D extensions of the level set tree and of the MSER representations, which are classic image analysis tools, and an intrinsic mesh segmentation method.

The underlying mathematical development focuses on an analysis of discrete differential operators acting on raw point clouds which have been proposed in the literature. By considering the asymptotic behavior of these operators on a smooth surface, a classification by their underlying curvature operators is obtained.

Of particular interest is the discrete operator which ends up being the numerical spinal chord of the chain. It is, in one sentence, *the iterated projection of each point on the local regression plane of its Euclidean neighborhood.* This iterated operator will be proved to be consistent with the mean curvature motion (the intrinsic heat equation). It therefore defines a remarkably simple and robust numerical *scale space* analysis. By this intrinsic heat equation (using its numerical reversibility), *all of the*

above mentioned different and difficult problems (point set orientation, raw point set triangulation and scan merging), usually solved by separated techniques, are solved in a unified framework. Inasmuch as they can handle large amounts of points, the existing methods have been compared to the new algorithms (see for example fig 1.1).

The next sections review one by one the chapters of the thesis and their contributions.

The thesis by chapters

State of the art (chapter 2)

This chapter gives an overview of the whole 3D surface processing field from the early acquisition to the final model rendering and post processing that can be done on the mesh models. It also reviews the multiple ways of acquiring 3D surfaces, acquisition methods being mainly divided into passive light acquisition methods and active light acquisition methods.

High precision point set orientation and meshing (chapter 3)

As stated above, the first goal is to visualize the initial raw point cloud including its very small details and its imperfections. All existing methods either smooth the shape or can simply not handle large data sets. Two major problems for 3D data set processing are solved using a new scale space implementation: the raw point set orientation, and the mesh construction. The result is a mesh whose vertices are **almost all** (in all examples more than 99.9%) of the input samples. It follows that the input data can be properly visualized.

(a) Picture of the Logo (b) Scale space mesh (c) Direct mesh (d) Poisson Mesh (e) MLS2 + BPA

Figure 1.1: Comparison between several meshing methods on a 1 cm high logo. See chapter 3 for more details

The main tool that will be used to solve these problems is the operator projecting each point into the local regression plane of its Euclidean neighborhood. An asymptotic interpretation of this operator will be proved: This operator is consistent with the mean curvature motion

$$\frac{\partial p}{\partial t} = H\mathbf{n}$$

where p is a point of the surface, and H and $\mathbf{n}$ are the mean curvature and normal to the surface at p. The iterated projection provides a perfect reversible numerical scale space, able to push the points toward a smooth surface or, once reversed, to transport back on the initial cloud a structure which was easily computed on the smoothed one.

Orienting a raw point set is not considered an easy problem: although normal directions are easy to compute, the ambiguity lies in finding a coherent orientation for the whole surface. Each point must be oriented consistently with its neighbors normals. This chapter proves the scale space method to work directly and reliably with the simplest implementation. The whole process is completely summarized in the following sentences. The point cloud is first smoothed by the scale space iteration, then the orientation of a seed point is chosen and its neighbors are oriented consistently with this normal. The orientation is then propagated from neighborhood to neighborhood. The final normals are transported back to the original positions of the points. Since sharp features are removed by scale space iterations, the propagation of normals is numerically straightforward.

Once a consistent orientation is found for the raw point cloud, the next step is to build a mesh from the scans. A mesh is the right shape model to generate efficient visualization. Chapter 3 shows that, again, the scale space method is self-sufficient to the task. The smoothed point cloud is easily meshed using any direct meshing algorithm such as the Ball Pivoting Algorithm ([BMR*99]). The resulting connectivity information (faces and edges of the mesh) can be transported back by reversing the scale space to the original point positions. As shown on figure 1.1, this method permits to recover the tiniest details of an object acquired by a high precision laser scanner, which are lost by other methods. Since this operator recovers the exact initial point positions, the final mesh is not smoothed and scan superposition artifacts can also be visualized.

These results will be published in *Scale Space Meshing of Raw Data Point Sets* [DMMSL09].

High fidelity scan merging (chapter 4)

Using the scale space meshing method, even the slightest scan misalignment is immediately visible (see fig 1.2(b)). Smoothing the shape, for example by a bilateral filter, would suppress such artifacts, yet it would also remove detail and texture and make the whole meshing method useless, since it was devised precisely to preserve and see exactly the initial data. A new way for merging different scans without smoothing them was therefore necessary.

(a) Input scans (b) Artefacts (c) Smooth base (d) Final merging

Figure 1.2: The scan merging problem and its solution . 38 scans cover the Tanagra head, represented in a different colors in 1.2(a).

Again, the scale space method proves sufficient to the task. Indeed, in a spirit similar to the low-high frequency decompositions in signal and image processing, the scale space operator can be used to decompose the shape into a low frequency base, the shape filtered by mean curvature motion (see fig. 1.2(c)), and a high frequency component (the scale space motion vectors). This decomposition can also be seen as a smooth base + height decomposition: the smooth base is the shape obtained after smoothing and the height function associates to each point its displacement vector.

Thus, it is shown that all scans can be decomposed separately but consistently into their smooth bases and their intrinsic high frequencies. But they can also receive a common base by applying the scale space to their joint cloud. Adding the scans high frequencies to the common smooth base yields a final merging where all details of all scans are preserved and merged together. This process preserves integrally the high frequencies, but slightly moves the smooth bases (see Fig 1.2(d)). The method proposed in this chapter is not a registration method: it only merges scans which are already well registered. But this process is essential to avoid very strong aliasing effect arising with even the slightest offset between scans.

The results of this chapter are published in *High Fidelity Scan Merging* [DMAL10].

Filling holes in scale space meshes (chapter 5)

This chapter makes a review of mesh hole filling methods and deduces an adaptation of the Liepa [Lie03] method to scale space meshes. Again, the method is based on the same scale space framework. It first detects and fills in the holes in the coarse scale mesh (the mesh built after the scale space iterations). In the regular scale space process, the backward operator should then be applied to get the fine scale mesh. Yet the backward scale space operator is not defined for the patch points. The patches can then be very conspicuous because of their smooth aspect.

To avoid this, a texture noise is added to the patch points in their normal directions. The texture properties are deduced from the motion amplitude of the first scale space iteration. Indeed, as stated above, the first iteration captures the local variations and this is what these patches lack. Implicitly the hypothesis is that the patches contain no feature but are textured as the rest of the shape.

The level set tree on meshes (chapter 6)

A third contribution deals with the problem of mesh segmentation and feature extraction. Most feature detection methods focus on crest lines (also called ridge and valley lines), yet this kind of feature detection requires the computation of an order 3 surface derivative and leads to a local detection similar to the edge detection paradigm in image analysis. To get a real segmentation method, a more global mesh feature is introduced : the level lines of a proper function defined on the mesh.

Here the analogy with image analysis is useful. Given a real function defined on its triangles, the mesh can be seen as the support of an image, the triangles as its pixels, and the function as the image values. Then level lines and level sets of this function can be considered as mesh features and be used to segment it. To extract the most significant image level lines, the Maximally Stable Extremal Region algorithm was extended to meshes, yielding a time-efficient reliable level set selection. But the obvious question is: which real function can be chosen to intrinsically characterize the mesh? It turns out that the mesh curvature is the simplest lower order local intrinsic function that can be deduced from the mesh itself. By (again!) the scale space method, this curvature can be computed at any scale of smoothing and transported back on the raw pixels. Thus the image defined on the raw mesh simply is its own curvature at some fixed scale. As this chapter shows, segmenting this image defined on the mesh yields a mesh segmentation into smooth parts and high curvature parts. The scale space operator therefore permits to define an intrinsic mesh segmentation.

Experimental results show the segmentation and line extraction working as well on mechanical parts (fig 1.3) as on fine arts objects (fig 1.4).

These results are published in *The Level Set Tree on Meshes* [DMAMS10].

A numerical analysis of raw point cloud smoothing (chapter 7)

For the sake of continuity in the exposition, this chapter comes late and is placed after the thorough description of the 3D scale space method and its application to the synthesis and analysis of raw point clouds.

But the chapter contains, nonetheless, the main mathematical results and the mathematical method to analyze the local discrete operators proposed in the literature and in the present thesis.

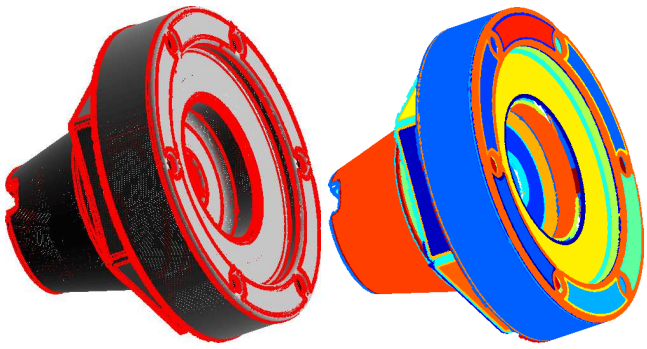


Figure 1.3: Line extraction (left) and segmentation (right) obtained by Mesh-MSER

(a) Picture of fragment 10g (b) Obtained Mesh (c) Mesh-MSER Selection

Figure 1.4: Example of the Mesh-MSER selection on an archaeological object

Indeed, the definition and mathematical analysis of a raw numerical scale space contributed in Chapter 3 leads to a general methodology for analyzing discrete methods. These methods attempt to compute differential operators on irregularly sampled surfaces by discrete schemes based on the local point cloud statistics. Many such methods have been proposed for meshed 3D data. However, computing directly differential operators on the raw point clouds as they are acquired (e.g.) by triangulation scanners is crucial because this can be done before any mesh resampling and can in particular bring decisive information into a meshing loop.

This chapter proposes a method to analyze and characterize these raw point cloud local operators. It reviews a half dozen basic discrete algorithms which have been proposed to compute discrete curvature-like shape indicators on raw point clouds. It shows that all of them can actually be analyzed mathematically in a unified framework by computing their asymptotic form when the size of the neighborhood tends to zero. Assuming that the underlying manifold is smooth enough, the asymptotic form of these operators is obtained in terms of the principal curvatures or of higher order intrinsic differential operators which characterize the discrete operators. This analysis, completed with numerical experiments, permits to classify the discrete raw point cloud operators, to rule out several of them, and to single out others.

Furthermore, by analyzing asymptotically two very simple moving least squares operators, namely the operator $MLS1(p)$ that projects each point p to the local regression plane (our favorite scale space operator) and the very common operator $MLS2(p)$ that projects each point onto the local degree 2 polynomial regression surface, it is shown that only the first is consistent with the mean curvature motion. More precisely, one has:

$$< MLS1(p) - p, \mathbf{n} >= \frac{r^2}{8}(k_2 + k_1) + O(r^3),$$

where r is the radius of the neighborhood used for the surface regression, k_1 and k_2 are the principal curvatures and $\mathbf{n}$ is the normal to the surface at p, oriented towards the concavity.

On the contrary, one has

$$< MLS2(p) - p, \mathbf{n} >= -\frac{r^4}{48}(3a_{04} + a_{22} + 3a_{40}) + O(r^5)$$

where $a_{40} = \frac{1}{4!}\frac{\partial^4 f}{\partial x^4}$, $a_{04} = \frac{1}{4!}\frac{\partial^4 f}{\partial y^4}$, $a_{22} = \frac{1}{4!}\frac{\partial^4 f}{\partial x^2 \partial^2 y}$ are the fourth derivatives of the graph function of the manifold in the intrinsic coordinates system at p and x, y are the coordinates along the principal directions.

These results will be published in [DM10]

Color cloud visualization (chapter 8)

Figure 1.5: An image and two views of its filtered color cloud

This last chapter shows that the scale space meshing method is able to triangulate even very noisy open surfaces. Such surfaces are given by another type of data: the color clouds. Color clouds are the set of RGB color values of the set of pixels of an image. They are contained in a 3D cube and can be filtered by the same scale space algorithm, revealing a 2D structure. The filtered data can then be meshed by scale space meshing and visualized. The idea is taken from [BLM10] where it was shown that color clouds have dimensionality 2. This chapter shows that the scale space

meshing algorithm can handle very different types of data. It also comforts the idea that color clouds are a set of potentially folded sheets in 3D. More importantly, the triangulation allows once again a precise visualization.

Publications linked to the thesis

- *Scale Space Meshing of Raw Data Point Sets*, Julie Digne, Jean-Michel Morel, Charyar Mehdi-Souzani, Claire Lartigue, currently under minor revision for Computer Graphics Forum. [DMMSL09]

- *High Fidelity Scan Merging*, Julie Digne, Jean-Michel Morel, Nicolas Audfray, Claire Lartigue, Computer Graphics Forum, vol 29, number 5, pp 1643-1651, Proceedings Symposium on Geometry Processing 2010. [DMAL10]

- *The Level Set Tree on Meshes*, Julie Digne, Jean-Michel Morel, Nicolas Audfray, Charyar Mehdi-Souzani, Proceedings of the Fifth International Symposium on. 3D Data Processing, Visualization and Transmission, Paris, France, 2010. [DMAMS10]

- *Neighborhood filters and the recovery of 3D information*, Julie Digne, Mariella Dimiccoli, Neus Sabater, Philippe Salembier, chapter in Handbook of Mathematical Methods in Imaging, Springer, to appear in 2010. [DDSS10]

- *A Numerical Analysis of Raw Point Cloud Smoothing*, Julie Digne, Jean-Michel Morel, to be submitted, 2010 [DM10]

- *Feature extraction from high-density point clouds: toward automation of an intelligent 3D contact less digitizing strategy*, Charyar Mehdi-Souzani, Julie Digne, Nicolas Audfray, Claire Lartigue, Jean-Michel Morel, proceedings of the CAD 2010 conference. [MSDA*10]

Outline of the thesis

The present thesis is divided as follows:

- Chapter 2 reviews state of the art work related to data acquisition, representation and extraction of characteristics.

- Chapter 3 describes the first applications of the new scale space: the robust orientation of surface point clouds and the construction of a high fidelity mesh whose vertices are the raw points.

- Chapter 4 introduces a method to merge scans to prevent the appearance of 3D aliasing due to scan misalignment.
- Chapter 5 describes a postprocessing step to detect and fill the holes in the built meshes.
- Chapter 6 describes the extension of the image level set tree theory to surface meshes.
- Chapter 7 reviews and analyzes mathematically the standard ways of computing the curvatures and curvature directions for raw point clouds.
- Chapter 8 describes another application of the projection filter: the filtering of color clouds extracted from color images.

Chapter 2

State of the Art

Contents

Abstract: This chapter reviews the whole 3D surface processing field from the early acquisition to the final model rendering and post processing that can be done on the mesh models.

2.1 3D Acquisition

There are multiple ways of acquiring 3D surfaces. We shortly explain how these data can be acquired. Acquisition methods are mainly divided into passive light acquisition methods and active light acquisition methods. Passive light acquisition methods do not project any light onto the object. This is the case of stereo acquisition for example. On the other side, active light scanner use light projection to acquire the geometry. There exist contact 3D scanners that acquire a surface by physical touch, yet here we will only consider non-contact 3D scanners.

2.1.1 Stereoscopy

Stereo acquisition is the principal passive light acquisition method: it creates a disparity map by considering two images in epipolar geometry and using the y axis difference between matching points to deduce a depth map. This is very useful in satellite imaging for creating digital elevation models for example (see chapter 6 fig 6.11)

2.1.2 Time-of-flight laser scanner

These range lasers measure the elapsed time between the emission of a pulse of light and the detection by a sensor of the light reflected by the surface. They are used for acquiring large objects. See Fig. 2.1 for an example of an object acquired by a range laser scanner. Time-of-flight scanners measure the surface one point at a time.

Figure 2.1: A Peugeot car acquired by a range laser scanner (acquisition made by Délégation Générale de l'Armement).

2.1.3 Triangulation laser scanner

Triangulation laser scanner are named this way because of the triangle formed by the laser emitter, the camera optic center and the laser impact point on the surface. Despite the name, triangulation laser scanners do not automatically produce a triangulation. They initially produce data points in a 3D coordinate system without any other information than the coordinates. Normal information could be deduced easily from the laser emitter position, yet being able to reliably process un-oriented points widens the application field of the algorithms. The most famous acquisition projects, *Digital Michelangelo* and *Forma Urbis Romae* (see section 2.2), have been done using triangulation laser scanner. For example, the laser system used for the David range acquisition " employed a 5 mW 660-nanometer laser diode, a 512 x 480 pixel CCD sensor, and a fixed triangulation angle" ([LPC*00]).

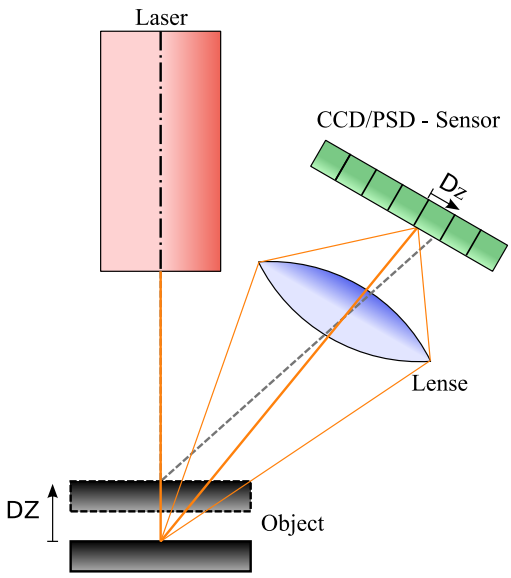


Figure 2.2: Scheme of a triangulation laser scanner acquisition system (copyright Wikipedia)

Efforts have been made to combine information of various kind during the acquisition process to improve the reconstruction quality:[NRDR05] combines normals computed by photometry and range images and [MYF99] integrates shape from shading and range images (see also [Jas97],[DSGV01]).

2.2 Some acquisition projects

Over the past years, some acquisition projects have been completed yielding new data sets for the geometry processing community. In particular, the Stanford Digital Michelangelo Project[1] aimed at building 3D models from some of Michelangelo's most important sculptures, including the David ([LPC*00], [BRM*02]).

The Stanford Forma Urbis Romae Project[2] dealt with a marble map of Rome designed in the early 3rd. century. This marble map was broken into several pieces, some of which were lost or came to us only through drawings. Trying to solve the jigsaw puzzle and putting back all the pieces together is a challenging task. The remaining pieces were therefore acquired by triangulation laser, which yielded another data set that is partly available for research ([KTN*06]). This data set will be used for experimentation in this thesis.

Most data used in this thesis come from the acquisition done by another laboratory: the Laboratoire Universitaire de Recherche en Productique Automatisée (LURPA). The next section shortly describes the acquisition system.

2.3 LURPA acquisition system

The LURPA acquisition system is a triangulation laser scanner with high precision (around $20\mu m$). It is composed of a granite table, a revolving arm and a laser scanner head (fig 2.3). The system projects a laser pencil on the surface whose position is captured by a CCD camera. The device is calibrated so that it can translate position coordinates given in the CCD coordinate system into the machine coordinate system. This process of registering scans together will be detailed in 2.5.1. Scans are registered to a very high precision but there is always a remnant offset which, as we shall see on the raw mesh, creates strong aliasing effects (see chapter 4). The output of this device is a set of unorganized un-oriented points given only by their 3D coordinates as can be seen on Fig. 2.3.

2.4 Representation of 3D data

The acquired shape comes usually as a set of points sampled with more or less precision on the object surface. A natural question is then to find a way to represent the surface in a more handy way than a list of 3D coordinates. The usual choice is to build a mesh: i.e., a set of connected triangles. This makes a piecewise linear approximation of the surface. The more vertices, the finer the approximation. Other surface representation include splines and NURBS surfaces

[1] http://www-graphics.stanford.edu/data/dmich-public/
[2] http://formaurbis.stanford.edu/docs/FURproject.html

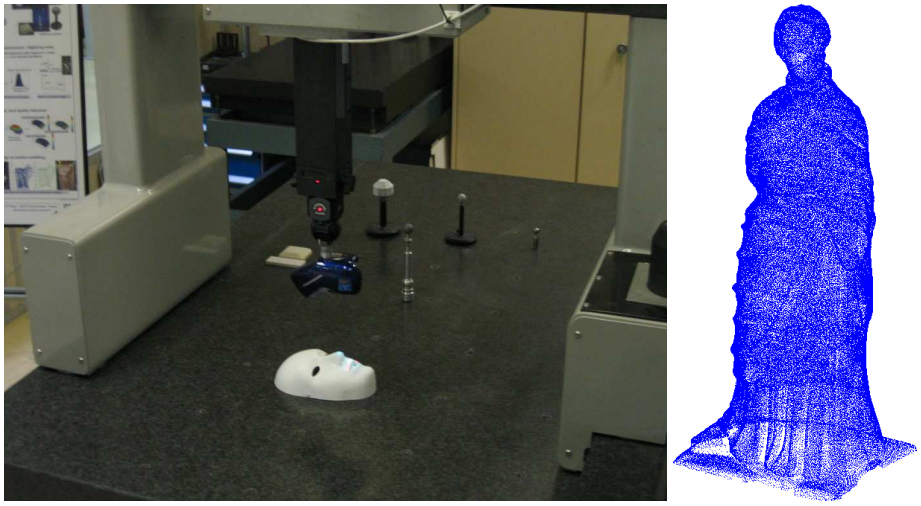


Figure 2.3: Picture of the LURPA scanner laser and its output: a raw point set

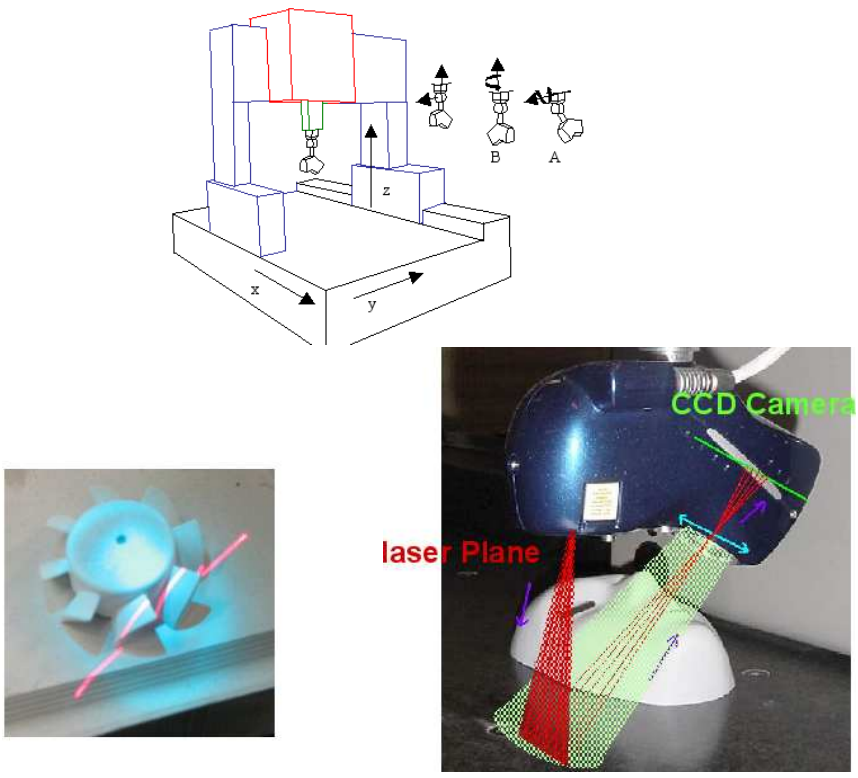


Figure 2.4: LURPA scanner device

([PT95],[MK98],[Pet98]). An alternative to meshes was presented in [LP03], where the shape is represented by a set of triangle fans. A triangle fan is a set of triangles sharing a vertex. This representation allows for faster computations because of the localness of the fans.

In this thesis, we show that meshes can be built on the original data points, thus loosing no information whatsoever on the raw data point set. In that way all scanning artifacts are revealed and can be carefully corrected, in a way that does not loose the initial point accuracy.

There are previous methods attempting to keep the point cloud information all along the process and to perform the final rendering on the point set itself, without meshing [LMR07], yet this is not our choice here. The whole theory of Moving Least Squares Surfaces [Lev03] also aimed at processing surfaces by local surface regression, without any previous meshing (see chapter 7). [MMS*07] proposed a meshless subdivision framework based upon the idea that point-based surface processing relies on intrinsic surface properties instead of intermediary representations. This idea of raw point processing is also at the bottom of our methods. Defining geodesics on point clouds has been investigated in [MS02] and [MS05a] where theoretical results are given for computing the distances on point clouds. [MS05b] and [MS04] use the Gromov-Hausdorff distances to compare manifolds given by point clouds. Another formulation was proposed in [BBK06]. [MS09] use geometric distance distributions on point clouds to recognize shapes.

Other works on point clouds include [BSW09] where an algorithm for building a Laplacian operator on point clouds is given and [MTSM10] which computes the visibility of a point cloud from a viewpoint.

The next section discusses how to go from an input raw point set to a mesh.

2.5 From a point cloud to a mesh

We assume that the scans are already registered, i.e., that all the scans are given in a global coordinate system, obtained by the device calibration or by previous registration algorithm (in the case of most data processed here the global coordinate system is given by the scanning device itself). Other acquisition systems do not necessarily provide this initial registration. This is why much work has been done on registering shapes to merge multiple view (fig 2.5).

2.5.1 Merging multiple views

The registration of multiple views is usually done by finding correspondences between the views and by finding a rigid or non-rigid transform mapping the set of points from the first view to the corresponding set of points in the second view.

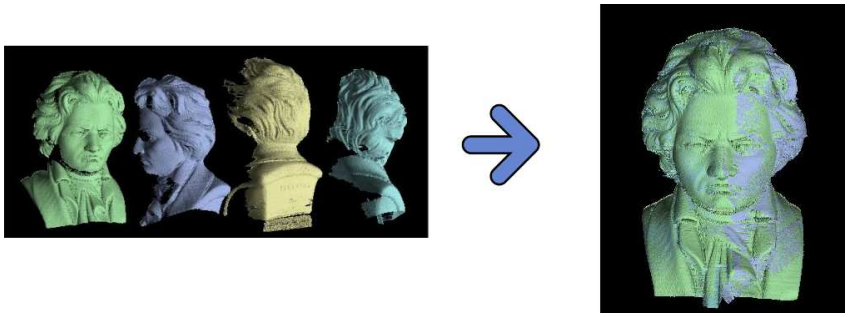


Figure 2.5: Registering multiple views (Image from David Laser Scanner Website)

Finding these initial correspondences requires a good shape descriptor. This topic has generated a huge literature. Famous shape descriptors include spin images ([JH97],[JH99], [Joh97]), snapshot descriptors ([Mal07]), regional point descriptors [FHK*04], spherical representations ([HID95], [KFR03]), harmonic maps ([ZH99]), point signatures ([CJ97],[YF99]) and 3D point's fingerprints ([SA01]). The heat kernel was also used to produce a point signature ([SOG09]).

Registration has also been studied in the shape matching context (e.g. to compare a shape to models in a database) [BBK*10], [BBK09], [BBBK09]. This is particularly difficult when dealing with deforming surfaces, for example, to recognize faces in spite of expression change. Specific surface descriptors had to be introduced ([BBK07], [BBK05a], [BBK05b],[BBK03]).

Once the initial registration has been found, the optimization is done by iteratively finding the registration that minimizes the distance between the overlapping parts of the scans and updating the point positions ([BM92],[RL01], [Hor87]).

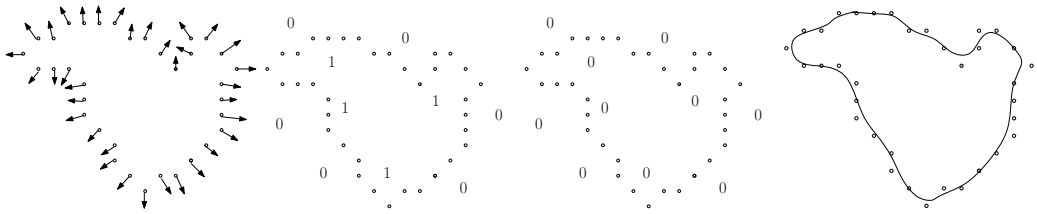
Since scans may contain warps, a rigid transform might not be enough to generate a good model, this is why non rigid transforms were considered ([ZZW*08], [CR03], [BR04], [BR07]). More specifically the transformation is not a rotation and translation anymore, but a thin plate spline allowing for distortions inside each scan.

Once scans are registered in the same coordinate system, points must be turned into a single mesh. The next subsection reviews the details of meshing.

2.5.2 Meshing

As soon as a set of points is given and we look for a suitable model, the interpolation/approximation dilemma comes in. Should the model include the initial data (i.e. the mesh vertices remain the input data set) or should it approximate the shape (the mesh is close to the initial points but does not necessarily contain the data)?

First, a mesh can be built by creating triangles between input samples. This is



(a) Initial oriented point set (b) Indicator function (c) Indicator gradient (d) Final surface

Figure 2.6: Level set method for reconstructing a surface.

the case of all Voronoi/Delaunay based reconstructions. Indeed by building the Delaunay tetrahedralization of a point set and labeling inside and outside tetrahedra, a set of frontier triangles can be extracted which are "on" the surface ([ACK01], [Boi84], [ABK98], [AB98]). Yet these methods do not handle properly large input point sets. Edelsbrunner and Mecke introduced the concept of α-*shapes* ([EM94], [Ede93], [BB97]). The α-shapes are based on the Delaunay diagram of the input samples and are used to build a sub-complex of the Delaunay simplicial complex. A triangle of the Delaunay triangulation is kept iff there exists a ball with empty interior or the complement of a ball of radius α that contains the whole pointset and such that the three vertices are on its frontier. The Ball-pivoting algorithm [BMR*99]) is based on this principle but avoids building the Delaunay triangulation. It simply rotates a ball of radius r on a set of points and builds triangles whenever the ball has three vertices on its frontier and none inside. For α-shapes as well as for ball-pivoting reconstruction, the problem is setting the ball radius which is a good compromise between detail loss and hole filling.

Over the past few years, another family of shape reconstruction methods has taken over the field. It is based on finding the level set of a function that more or less corresponds to the distance to the underlying shape (see Fig 2.6). Some methods assume that the points are consistently oriented, and use various function family to approximate the signed distance field: Fourier basis or radial basis functions for example ([HDD*92], [Kaz05], [KBH06], [ACSTD07], [CBC*01], [OBA05]), while more recent methods deal with unoriented point sets ([MDGD*10]). An interesting method uses the Voronoi diagram in the level set framework: the distance function is approximated by radial basis functions centered at the voronoi cell centers ([SAAY06]). Once the distance function is built, the level set is extracted using the Marching Cubes algorithm ([LC87]) or its extension ([KBSS01]). Other possibilities include [WH94], [CA97] and [LGS06].

2.5.3 Remeshing

Meshes built from raw scans might have a high number of vertices. Therefore, reducing the number of vertices (compressing the mesh data) is of crucial importance for practical uses. Though remeshing will not be addressed in this thesis (it will only be used in the hole filling process), we summarize briefly the remeshing field (a review of remeshing techniques can be found in [AUGA05]). Decreasing the number of vertices and triangles while keeping the shape as close as possible to the original was investigated in [HDD*93], [PGK02] and [CSAD04] among others. The aim of remeshing can also be to improve the mesh regularity ([AMD02],[BK04], [AVDI03]). Usually the remeshing is done to satisfy a remeshing criterion and many such criteria exist (see [ABE99]). Another popular remeshing topic is to transform a triangle mesh into a quadrangle mesh. Indeed quadrangles are attractive because of their tensor-product like nature, which is useful for mapping textures (e.g.) [ACSD*03],[AUGA05],[SLI98] or NURBS patch fitting in reverse engineering. Other methods have been proposed to redistribute vertices according to a density function ([PC06], [PC05], [PC03]).

In a related domain, [LCOLTE07] proposed a projection algorithm to project an arbitrary point set onto an input point-set which can be used to resample surfaces by point sets.

2.6 Rendering

Once the mesh is built, the object must be rendered to be visualized. This can be done by *Ray Tracing* methods. The idea is to trace a ray from the optic center of a virtual camera through each pixel in a virtual screen, and computing the color of the object that this ray intersects (see fig 2.7). It requires computing mesh-ray intersections which can be very slow in case of large meshes. Models can be textured and various types of lights can be set to get realistic renderings.

2.7 Post-processing and line detection

When surfaces are built, their geometry can be analyzed. For example, we might want to extract high curvature lines. This is always a very delicate part since high curvature lines, crest and ridge lines would actually need computing zero crossing of principal curvatures derivatives. This is an order 3 derivative of a surface we know only by a few sampled points. In case of a very smooth surface it could actually work, but it proves very unstable. Chapter 6 will introduce a new, global way for extracting geometry from a shape.

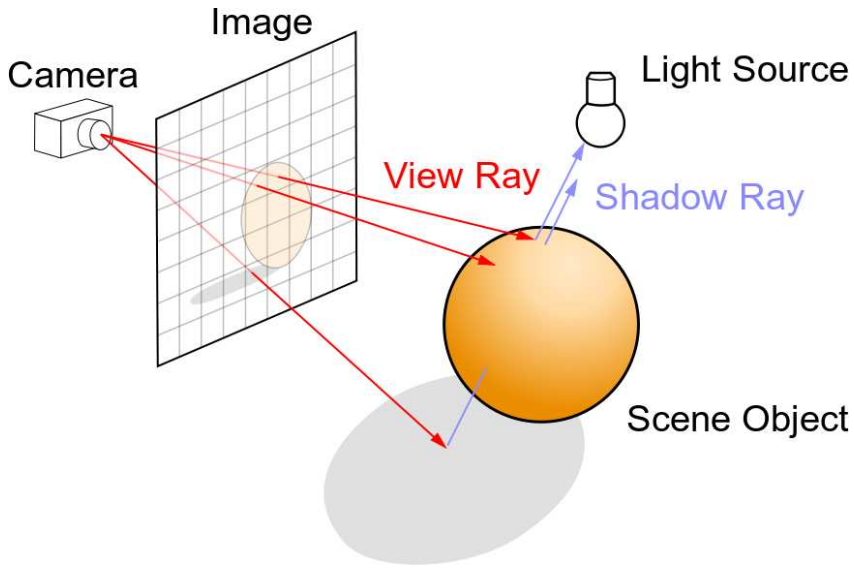


Figure 2.7: Scheme of ray tracing (copyright Henrik, Wikipedia)

In accordance with the edge detection paradigm in image processing, it is common to perform a 3D shape analysis by extracting the crest lines (the real edges) on meshes or point clouds. Ridge lines are the loci of points where the maximal curvature takes a positive maximum along its curvature line. Valley lines are the loci of points where the minimal principal curvature attains a negative minimum along its curvature line. These points can be linked to form lines (see among others [OBS04], [BA05], [LFM96], [YBS05], [SF04]). Most methods use a quadric or polynomial regression. In [GWM01], the lines are detected by neighborhood covariance analysis. Indeed, from a point neighborhood, the centroid and centered covariance can be computed. Comparing the ratios of the covariance matrix eigenvalues gives the geometry of the neighborhood (see also [MOG09]). In [HMG00], edges of a mesh are first classified according to their importance (this importance is an increasing function of the adjacent faces angle).

A multiscale approach was proposed in [PKG03]. Nearby feature points are first detected. In the neighborhood of these points surfaces are fitted, and depending on the number of fitted surfaces, points are projected to the nearest surface. Intersection points of these surfaces are finally classified as edge or corner points. By increasing the processing radius, one could track feature lines and keep only the ones at a given scale. Though dealing with scales, this method does not introduce a scale space framework. A similar idea for points classification and point projection was used in [DIOHS08].

Although these papers introduce a ridge/valley line detection, none of them proposes a ridge and valley segmentation. In [IFP95] the idea was suggested,

though: points lying near ridges or near valleys were labeled and this labeling was used to obtain a better rendering of the ridge and valley lines. But crest lines as defined by these methods require the computation of degree three surface derivatives.

Chapter 6 focuses on other interesting and well defined line features: namely the curvature level lines and level sets, analogous to the image gray level lines. Of particular interest are the zero-crossings of the curvature, which are technically similar to the zero-crossings of the Laplacian in image processing. These zero-crossings define inflexion lines, easy to compute from the raw data point set. They reliably segment the surface into ridges and valleys.

2.8 The difference between Image Processing and 3D surface processing

Both subjects are very related and indeed share common ideas. Nevertheless, there is a major difference: there is no equivalent of the Shannon sampling theory for 2D surfaces embedded in three dimensions, no Fourier analysis, and no notion of frequency domain.

This is why so many algorithms cannot be easily adapted to surfaces. For example, the very powerful non local means algorithm [BCM05] was considered for adaptation to meshes by Yoshizawa et al. ([YBS06]) yet the sampling problem was not really handled. It was once more adapted in [WZZY08], more properly so that neighborhoods were better defined, yet it made the whole algorithm much more complicated. As will be seen in Chapter 6, however, a (non linear) high frequency-low frequency decomposition of the surface is actually possible thanks to the scale-space method developed in the present thesis. It provides a way to create for each surface a smooth base on which the high frequency component can be defined as a refinement field. This (simple) decomposition will be used for both the processing (chapter 6) and the analysis of the surface (chapter 4).

Chapter 3

Scale Space Meshing of Raw Data Point Sets

Contents

Abstract: This chapter develops a scale space strategy for orienting and meshing exactly high resolution very large raw data point sets. The scale space is based on the intrinsic heat equation, also called mean curvature motion (MCM). A simple iterative scheme implementing MCM directly on the raw points is described, and a mathematical proof of its consistency with MCM is given. Points evolved by this MCM implementation can be trivially backtracked to their initial raw position. Therefore, both the orientation and mesh of data point set obtained at a smooth scale can be transported back on the original. The gain in visual accuracy is demonstrated on archaeological objects by comparisons with other meshing methods. The method permits to visualize raw data point sets coming directly from a scanner, and to put in evidence all scanning artifacts (aliasing, holes,...), thus permitting to correct them (chapters 5, 4) and to evaluate the quality of the correction. The robustness of the method will also be demonstrated on very noisy point clouds coming from color histograms (chapter 8).

3.1 Introduction

A growing number of applications involve creating numerical models for existing objects acquired by triangulation laser scanner or other devices. Commercial scanners can directly produce a direct triangulation of points sampled on the surface, but this triangulation is derived from a raw set of points with no connectivity information. Only raw input data will be considered here, namely sets of unorganized and non-oriented points given by their x, y, z coordinates. The proposed method orients and meshes directly the complete raw data set, thus allowing for the visualization of the finest surface details, and an accurate delineation of the scanning holes. The processed point data have a typical acquisition error of 20μ, allowing in principle to recover the finest texture and details.

The main tool introduced here is a raw data set point smoothing operator consistent with the intrinsic heat equation. The intrinsic heat equation, or mean curvature motion (MCM), is the simplest intrinsic method to smooth out a surface. The mean curvature motion writes

$$\frac{dp}{dt} = H\mathbf{n} \tag{3.1}$$

where H is the mean curvature at p (whose sign depends on the normal orientation), and $\mathbf{n}$ the normal. This motion will be given a robust implementation working directly on raw data points, which can be summarized in few words: *it is the iterated projection of each point on the regression plane of its radial neighborhood.*

Mathematical and experimental arguments will show that this iterated planar regression consistently implements the MCM and actually permits to compute an accurate denoised curvature. Indeed, Theorem 9 states that, by these iterations, each raw data set point moves forward at the speed of the surface mean curvature in the direction of the surface normal.

By the iterated projection algorithm each initial raw data point can be tracked forward and backward in the surface smoothing process. As a consequence, the surface structure established at a smooth scale can be transported back on the raw data set point. This back transportation yields a topologically faithful orientation at each raw point, and subsequently a mesh whose vertices are almost all raw data points. It also permits an accurate detection of holes in the raw data, useful for further scanning attempts. Comparative experiments will show that a direct meshing gives poor results, while the back transported mesh allows for the uttermost accurate rendering of the surface, the mesh vertices being more than 99% of all initial raw points. Obviously, such a complete mesh is not economical, but it permits an accurate rendering of fine art or archaeological pieces at 20μ precision and a detection by visual inspection of the tiniest scanning defects.

The use of the mean curvature motion, forward and backward, is a direct 3D extension of the scale space paradigm in image processing introduced in the founding Witkin paper [Wit83]. It consists of applying the heat equation $\frac{\partial u}{\partial t} = \Delta u$ to the image u, which entails a rapid image simplification. The main image features (for example the edges) are detected at a coarse scale (large t) and then tracked back to their fine scale position. The next subsection reviews the methods computing curvatures and normals on raw data.

3.1.1 Building a mesh

Given an initial *oriented* point cloud, most meshing methods start by defining a signed distance field to the inferred surface [HDD*92],[KBH06]. The signed distance function can be estimated at any point by computing the distance between the point and the regression plane of its k-nearest neighbors [HDD*92]. Since the neighbors are assumed previously oriented, the sign of this distance is straightforward. Other successful methods approximate the distance function using its decomposition on a local radial basis functions [KBH06]. Once the distance function is defined, extracting the surface corresponds to extracting the zero level set of the distance function. This can be done using the marching cubes algorithm [LC87] or any other other contouring algorithm.

These methods yield meshes that approach well the shape, *but the approximation entails an implicit surface smoothing and the loss of fine texture.* Acquisition holes are also filled in by those methods, the signed distance function giving a natural close up of the surface. Nonetheless, for scanning applications, the acquisition

holes should be detected rather than filled in. The smoothing can be desirable if there are noise and scanning artifacts. However, in the cases we shall examine, texture actually dominates noise. A guarantee that no detail is lost is granted only when almost all raw data points are mesh vertices. [AMD02] introduces a remeshing method based on mappings of the mesh onto a plane. Meshing a planar projection will also be used here, but this projection will only be local.

In a way the scale space meshing method is not far from [GKS00], where the triangulation is found by locally projecting the points onto a regression plane and performing a 2D triangulation. Our method will also consider meshing a simpler point set yet it uses a 3D triangulation method and is done in the scale space framework defined below.

3.1.2 Raw data point set processing

Yet, it has been termed impossible to mesh directly the raw data point set. The literature has therefore considered more and more sophisticated smoothing and interpolation methods. The "Moving Least Square Surface" (MLS) introduced in [Lev03] is defined as the set of stationary points of an operator projecting each raw point on a local quadratic regression of the surface. The order n MLS algorithm estimates at each point a degree n polynomial surface from a set of weighted neighbors. The obtained surface can be used to project the point on the MLS surface, or to sub-sample the surface by removing step by step the points with least influence [ABCO*03]. Variations of the MLS algorithm for denoising point sampled surfaces and preserving edges were proposed in [FCOS05], [GTE*06],[OGG09],[LCOL07], [GG07]. Interpolatory point set surfaces can be achieved using a singular weighting function ([OGG09], [AA09],[SOS04]), but extracting the isosurface via marching cubes will loose the input point positions

At first sight applying MCM to a data point set requires the separate computations of the surface intrinsic Laplacian (mean curvature) and of the normal. For meshes, the standard discretization of the Laplacian operator is done through the cotangent formula [MDSB02]. For point clouds, [BSW09] proposed the construction of a laplacian operator for functions defined on point clouds (yet no result on real noisy shapes was presented). In [PKG06], the curvature is either estimated by a polynomial regression or by projection on a fitted least square surface (in other terms, by MLS). The reverse operator is built by storing the displacements of each point at each step. A similar scale space approach will be used here, but with quite different scopes: in [PKG06], the proposed applications were shape morphing and shape editing.

In [UH08], another raw data point set MCM discretization was proposed. The surface Laplacian is computed by building an operator A_θ at each point position and for every direction θ in the tangent plane. A_θ moves a point p proportionally

to the curvature H_θ of the section curve in direction θ. By integrating over θ, it yields a mean curvature motion.

3.1.3 Computing curvatures

Computing the principal curvatures reliably on a given surface is crucial for various applications, in particular the anisotropic filtering preserving sharp edges [HP04], [MDSB02], or the sampling methods adapting the density to the surface curvatures [PGK02]. On meshes, the curvature estimation problem has already been investigated in [MDSB02] where the cotangent formula is proven and extended. [Tau95a] derived an analytic expression for estimating the directional curvatures in the edge directions. In [Rus04], [TRZS04], the tensor curvature was estimated on each face of a mesh surface. Other mesh curvature computation techniques include the use of the normal cycle theory [CSM03]. For a summary and comparison of mesh curvature estimation methods, see [MSR07]. It is also possible to estimate curvatures by building curves contained in the surface and passing through the considered point [Tan05].

To determine the curvature of a given point, direct methods fit a surface (a polynomial or a quadric) locally to each neighborhood and then compute the fundamental forms in their explicit form. This permits to compute the Weingarten map whose eigenvalues and eigenvectors are the principal curvatures and principal directions ([LFM96] among others). In [BC94] the principal curvatures are computed from an oriented raw data set without surface fitting by expressing the fundamental forms of a 3D surface as covariance matrices. Indeed, the covariance matrix of the point normals projected on the regression plane yields the principal curvatures and their directions. Other approaches avoiding surface regression include the computation of integral invariants [PWHY09],[PWY*07]. They are based on the idea that differentiation is not robust in a discrete and potentially noisy data set, whereas integration is much more resistent to noise. The proofs link the computation of the area of the intersection of the surface with a ball to the principal curvatures. Another possibility is to adapt the curvature estimation of [Tau95a] to the case of point clouds, like in [LP05]. Instead of considering the edge direction, since no edge information is given for the point cloud, they consider all directions from the center point to one of its neighbors. MLS surfaces were also used to derive analytic expressions for the curvatures of point set surfaces [YQ07]. As far as meshes are concerned, a comparison of various curvature estimations can be found in [SMS*03].

Mathematical results are given in chapter 7 proving the consistency of the proposed scale space algorithm. This chapter is divided as follows: section 3.3 analyzes the discretization problem. Sections 3.4, 3.5 describe the two main applications of the scale space: a point cloud orientation method and a faithful mesh construction

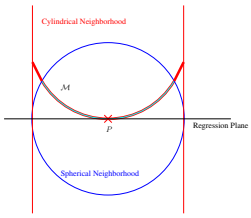


Figure 3.1: Comparison between cylindrical and spherical neighborhoods

for the raw data set. Comparative experiments are presented in section 3.6.

3.2 Continuous theory

This section investigates a new way of implementing the mean curvature motion by the iteration of a planar surface regression. The surface $\mathcal{M}$ supporting the data point set is assumed to be at least C^2. The samples on the surface $\mathcal{M}$ are denoted by $\mathcal{M}_S$.

Let $p(x_p, y_p, z_p)$ be a point of the surface $\mathcal{M}$. At each non umbilical point p, consider the principal curvatures k_1 and k_2 linked to the principal directions $\mathbf{t}_1$ and $\mathbf{t}_2$, with $k_1 > k_2$ where $\mathbf{t}_1$ and $\mathbf{t}_2$ are orthogonal vectors. (At umbilical points, any orthogonal pair $(\vec{t_1}, \vec{t_2})$ can be taken.) Set $\mathbf{n} = \mathbf{t}_1 \times \mathbf{t}_2$ so that $(\mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$ is an orthonormal basis. The quadruplet $(p, \mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$ is called the local intrinsic coordinate system. In this system we can express the surface as a C^2 graph $z = f(x, y)$. By Taylor expansion,

$$z = f(x, y) = \frac{1}{2}(k_1 x^2 + k_2 y^2) + o(x^2 + y^2). \tag{3.2}$$

Notice that the sign of z depends on the arbitrary surface orientation.

3.2.1 Spherical neighborhoods vs cylindrical neighborhoods

Consider two kinds of neighborhoods in $\mathcal{M}$ for p defined in the local intrinsic coordinate system $(p, \mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$:

- a neighborhood $\mathcal{B}_r = B_r(p) \cap \mathcal{M}$ is the set of all points m of $\mathcal{M}$ with coordinates (x, y, z) satisfying $(x - x_p)^2 + (y - y_p)^2 + (z - z_p)^2 < r^2$
- a cylindrical neighborhood $\mathcal{C}_r = C_r(p) \cap \mathcal{M}$ is the set of all points $m(x, y, z)$ on $\mathcal{M}$ such that $(x - x_p)^2 + (y - y_p)^2 < r^2$.

For the forthcoming proofs the cylindrical neighborhood will prove much handier than the spherical one. The next technical lemma justifies its use.

Lemma 1. *Integrating on $\mathcal{M}$ any function $f(x,y)$ such that $f(x,y) = O(r^n)$ on a cylindrical neighborhood $\mathcal{C}_r$ instead of a spherical neighborhood $\mathcal{B}_r$ introduces an $O(r^{n+4})$ error. More precisely:*

$$\int_{\mathcal{B}_r} f(x,y)dm = \int_{x^2+y^2<r^2} f(x,y)dxdy + O(r^{4+n}). \tag{3.3}$$

Proof. The surface area element of a point $m(x,y,z(x,y))$ on the surface $\mathcal{M}$, expressed as a function of x, y, dx and dy is $dm(x,y) = \sqrt{1+z_x^2+z_y^2}dxdy$. One has $z_x = k_1 x + O(r^2)$ and $z_y = k_2 y + O(r^2)$. Thus

$$dm(x,y) = \sqrt{(1+k_1^2x^2+k_2^2y^2+O(r^3))}dxdy$$

which yields

$$dm(x,y) = (1+O(r^2))dxdy. \tag{3.4}$$

Using (3.4), the integrals we are interested in become

$$\int_{\mathcal{C}_r} f(x,y)dm = (1+O(r^2))\int_{\mathcal{B}_r} f(x,y)dxdy; \tag{3.5}$$

$$\begin{aligned}\int_{\mathcal{B}_r} f(x,y)dm &= (1+O(r^2))\int_{\mathcal{C}_r} f(x,y)dxdy \\ &= (1+O(r^2))\int_{x^2+y^2<r^2} f(x,y)dxdy.\end{aligned} \tag{3.6}$$

This last form is amenable to analytic computations. Consider polar coordinates (ρ,θ) such that $x = \rho\cos\theta$ and $y = \rho\sin\theta$ with $-r \le \rho \le r$ and $0 \le \theta \le \pi$. Then for $m(x,y,z)$ belonging to the surface $\mathcal{M}$, we have $z = \frac{1}{2}\rho^2(k_1\cos^2\theta + k_2\sin^2\theta) + O(r^3)$. Fixing θ we obtain a curve with equation $z = \frac{1}{2}\rho^2 k(\theta) + O(r^3)$, where $k(\theta) = k_1\cos^2\theta + k_2\sin^2\theta$. The condition that (x,y,z) belongs to the neighborhood $\mathcal{B}_r(p)$ can therefore be rewritten as $\rho^2+z^2<r^2$, that is

$$\rho^2 + \frac{1}{4}k(\theta)^2\rho^4 < r^2 + O(r^5)$$

Computing the boundaries $\pm\rho(\theta)$ of this neighborhood yields $\rho(\theta)^2 + \frac{1}{4}k(\theta)^2\rho(\theta)^4 - r^2 + O(r^5) = 0$. Thus

$$\rho(\theta)^2 = \frac{-1+\sqrt{1+k(\theta)^2(r^2+O(r^5))}}{\frac{1}{2}k(\theta)^2}.$$

This yields $\rho(\theta) = r - \frac{1}{8}k(\theta)^2 r^3 + O(r^3)$. We shall use this estimate for the error term E appearing in

$$\begin{aligned}\int_{\mathcal{B}_r} f(x,y)dxdy &= \int_{[0,2\pi]}\int_{[0,\rho(\theta)]} f(x,y)\rho d\rho d\theta \\ &= \int_{[0,2\pi]}\int_{[0,r]} f(x,y)\rho d\rho d\theta - E \\ &= \int_{\mathcal{C}_r} f(x,y)dxdy - E,\end{aligned}$$

with $E =: \int_{[0,2\pi]}\int_{[\rho(\theta),r]} f(x,y)\rho d\rho d\theta$. Thus

$$|E| \leq \frac{\pi}{4}\sup_{x^2+y^2\leq r^2} |f(x,y)|k(\theta)^2 r^4,$$

which yields $|E| \leq \frac{\pi|k_1|^2}{4}\sup_{x^2+y^2\leq r^2}|f(x,y)|r^4$. In particular if $f(x,y) = O(r^n)$, then $|E| \leq O(r^{4+n})$. Finally we have

$$\int_{\mathcal{B}_r} f(x,y)dxdy = \int_{\mathcal{C}_r} f(x,y)dxdy + O(r^{4+n}). \tag{3.7}$$

Combining (3.5), (3.6) and (3.7) yields (3.3). □

3.2.2 Curvature estimation

Theorem 1 states that projecting a point onto the neighborhood barycenter approximates the mean curvature motion. We shall discuss later on why it cannot be used for implementing the mean curvature motion.

Theorem 1. *In the local intrinsic coordinate system, the barycenter of a neighborhood $\mathcal{B}_r$ where p is the origin of the neighborhood has coordinates $x_o = o(r^2)$, $y_o = o(r^2)$ and $z_o = \frac{Hr^2}{4} + o(r^2)$, where $H = \frac{k_1+k_2}{2}$ is the mean curvature at p.*

Proof. By Lemma 1 applied to the numerator and denominator of the following fraction, we have

$$\begin{aligned}z_o &= \frac{\int_{\mathcal{B}_r} z dm}{\int_{B_r} dm} = \frac{\int_{x^2+y^2<r^2} z(x,y)dxdy + O(r^5)}{\int_{x^2+y^2<r^2} dxdy + O(r^3)} \\ &= \frac{\int_{x^2+y^2<r^2}\left[\frac{1}{2}(k_1x^2+k_2y^2) + o(x^2+y^2)\right]dxdy}{\int_{x^2+y^2<r^2} dxdy} + O(r^3) \\ &= \frac{1}{2\pi r^2}\int_{\rho=0}^{r}\int_{\theta=0}^{2\pi}\rho^2(k_1\cos^2\theta + k_2\sin^2\theta)\rho d\rho d\theta + o(r^2) \\ &= \frac{r^2}{8\pi}(k_1\pi + k_2\pi) + o(r^2) = \frac{Hr^2}{4} + o(r^2).\end{aligned}$$

A similar but simpler computation yields the estimates of x_o and y_o. □

3.2.3 Surface motion induced by projections on the regression plane

The main tool of the proposed scale space will be simple projection of each surface point p on the local regression plane. This PCA regression plane is defined as the plane orthogonal to the least eigenvector of the centered local covariance matrix, and passing through the centroid of the neighborhood. The projection of p on this plane will be called p'. The next lemma compares the normal to the PCA regression plane with the normal to the surface, $\mathbf{n}$.

Lemma 2. *The normal* $\mathbf{v}$ *to the PCA regression plane at* $p \in \mathcal{M}$ *is equal to the surface normal at point* p*, up to a negligible factor:* $\mathbf{v} = \mathbf{n} + O(r)$.

Proof. The local PCA regression plane of point p is characterized as the plane passing through the barycenter of the neighborhood $\mathcal{B}_r$ and with normal $\mathbf{v}$ minimizing:

$$I(\mathbf{v}) = \int_{\mathcal{B}_r} |\langle \mathbf{v}, pm \rangle|^2 dm \text{ s.t. } \|\mathbf{v}\| = 1$$

Denoting by (v_x, v_y, v_z) the coordinates of $\mathbf{v}$,

$$I(\mathbf{v}) = \int_{\mathcal{B}_r} (v_x x + v_y y + v_z \frac{1}{2}(k_1 x^2 + k_2 y^2) + o(r^2))^2 dxdy.$$

Considering the particular value $\mathbf{v} = (0, 0, 1)$ shows that the minimal value I_{min} of $I(\mathbf{v})$ satisfies $I_{min} \leq O(r^6)$. In consequence the minimum (v_x, v_y, v_z) satisfies $v_x \leq O(r)$ and $v_y \leq O(r)$. Thus $v_z \geq 1 - O(r)$ and therefore $\mathbf{v} = \mathbf{n} + O(r)$. □

By Lemma 2, projecting p onto the regression plane induces a motion which is asymptotically in the normal direction: $p'p$ is almost parallel to $\mathbf{n}$. The simple projection of each surface point p onto its local regression plane approximates a 3D scale space (mean curvature motion) as shown in the next theorem.

Theorem 2. *Let* T_r *be the operator defined on the surface* $\mathcal{M}$ *transforming each point* p *into its projection* $p' = T_r(p)$ *on the local regression plane. Then*

$$T_r(p) - p = \frac{Hr^2}{4}\mathbf{n} + o(r^2). \tag{3.8}$$

Remark More extensive formal calculations (too long to be included here) prove that the error term in Theorems 1 and 2 is in fact $O(r^4)$.

Proof. By Theorem 1 the barycenter o of $\mathcal{B}_r$ has local coordinates $\overrightarrow{po} = (o(r^2), o(r^2), \frac{Hr^2}{4} + o(r^2))$. On the other hand $\overrightarrow{pp'}$ is proportional to the normal to the regression plane, $\mathbf{v}$. Thus by Lemma 2 $\overrightarrow{pp'} = \lambda(O(r), O(r), 1 - O(r))$. To compute λ, we use the fact that p' is the projection on the regression plane of p, and that o belongs to this plane by definition. This implies that $\overrightarrow{pp'} \perp \overrightarrow{op'}$ and therefore

$$\lambda^2 O(r^2) + \lambda(1 - O(r))(H\frac{r^2}{4} + o(r^2) + \lambda(1 - O(r))) = 0,$$

which yields $\lambda = \frac{Hr^2}{4} + o(r^2)$ and therefore

$$\overrightarrow{pp'} = (O(r^3), O(r^3), \frac{Hr^2}{4} + o(r^2)) = \frac{Hr^2}{4}\mathbf{n} + o(r^2).$$

□

3.3 The discrete algorithm

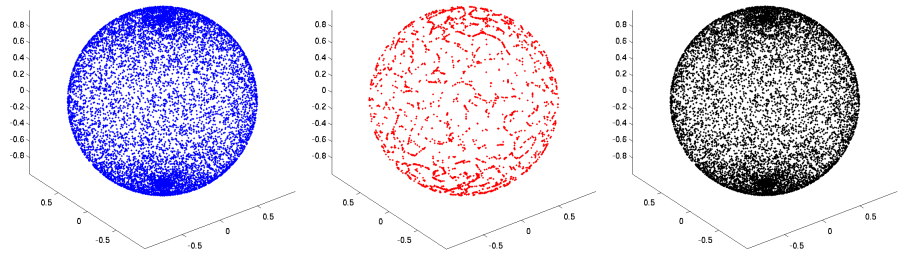
The previous theorems assume that the surface is a uniform Lebesgue measure. A constant sampling density is therefore necessary. This constant density will be approximated on discrete data by weighting each point by a weight inversely proportional to its initial density. More precisely, let p be a point and $\mathcal{B}_r = \mathcal{M}_s \cap B_r(p)$. Each point m should ideally have a weight $0 \leq w(m) \leq 1$ such that $\sum_{m \in \mathcal{B}_r} w(Q) = 1$. This amounts to solve a huge linear system. For this reason, we shall be contented with ensuring $\sum_{m \in \mathcal{B}_r} w(Q) \simeq 1$ by taking $w(p) = \frac{1}{\sharp(\mathcal{B}_r)}$, as proposed in [UHo8]. Let o be the weighted barycenter of this neighborhood. In $\mathbb{R}^3$, the coordinates are written with superscripts, e.g. the coordinates of a point u are (u^1, u^2, u^3). Thus, for $i = 1, 2, 3$, $o^i = \frac{1}{\sum_{m \in \mathcal{B}_r} w(m)} \sum_{m \in \mathcal{B}_r} w(m) m^i$. The centered covariance matrix $\Sigma = (m_{ij})_{i,j=1,\cdots,3}$ is defined as $m_{ij} = \sum_{m \in \mathcal{B}_r} w(m)(m^i - o^i) \cdot (m^j - o^j)$ for $i, j = 1, 2, 3$. Let $\lambda_0 \leq \lambda_1 \leq \lambda_2$ be the eigenvalues of Σ with corresponding eigenvectors v_0, v_1, v_2. For $k = 0, 1, 2$,

$$\lambda_k = \sum_{m \in \mathcal{B}_r} w(m) \langle (m - o), \mathbf{v}_k \rangle^2. \tag{3.9}$$

Each eigenvalue gives the variance of the point set in the direction of the corresponding eigenvector. Since v_1 and v_2 are the vectors that capture most variations, they define the PCA regression plane. The normal $\mathbf{v}$ to this plane is the direction $\mathbf{v}$ minimizing $\sum_{m \in \mathcal{B}_r} w(m) \langle (m - o), \mathbf{v} \rangle^2$.

Effectiveness of Theorems 1 and 2. Both Theorems permit *a priori* to implement the mean curvature motion on the raw data point set *without any previous orientation*. Nevertheless, the numerical application of these theorems depends on the

assumption that a uniform Lebesgue measure on the surface is well represented by a uniform sample density. This is not true for the barycenter method of Theorem 4. Iterating the barycenter method with a small neighborhood and a slightly varying sample density leads to a local clustering of the samples. Indeed, the barycenter method provokes a normal motion, but also a non negligible tangential motion to the surface. This motion is precisely the one used in the Mean Shift method [Che95] for data clustering. This undesirable clustering effect is illustrated in fig. 3.2. Even though the point distribution on the sphere is probabilistically uniform, local clustering occurs by taking local barycenters. In contrast, for the projection filter there is no observable tangential shift on the right image of fig. 3.2. Theorem 9 is in that case consistent with its numerical implementation. This follows from the obvious fact that any (non aligned) irregular sampling of a plane permits to *exactly* recover the plane by linear regression.



(a) Original samples on a sphere (b) 4 barycenter iterations (c) 4 projection filter iterations

Figure 3.2: Comparison of the iterated barycenter and of the iterated projection filter on a randomly sampled sphere. Both motions are consistent with the mean curvature motion, but the iterated barycenter provokes clustering. See text for an explanation.

Back propagation A normal motion by mean curvature can be defined for every point p_0 on the initial surface as a solution of (3.1) ($\frac{dp}{dt} = H\mathbf{n}$) considered as an ordinary differential equation with initial point p_0. Thus, the backward scale space is trivial, provided the forward MCM implementation actually implements the evolution of each raw data set point p_0. Let us consider a point p_t and its evolution p_{t+1} at steps t and $t+1$. Now, we can build the sequence $d_p(t) = p_{t+1} - p_t$ and the reverse scale space operator $\mathcal{P}_t^{-1}(p_{t+1}) = p_{t+1} - d_p(t)$, this operator allows to go backward in the scale space evolution from step $t+1$ to 0. This is exactly the construction proposed in [PKG06]. If we only need to go from step t to the initial data 0, without any intermediate step, the operator is even simpler to build, since we only need to store for each point p_t its initial position $\mathcal{P}_t^{-1}(p_t) = p_0$. This reverse

scale space operator will be called *back propagation, or back transportation*.

3.3.1 Higher order regression surfaces

The authors of [CP03] proved that a degree n polynomial fitting estimates all $k^{th} order$ differential quantities to accuracy $O(h^{n-k+1})$. In [PKG06] an order 2 Moving Least Squares (MLS2) method projecting the point onto the locally fitted least squares surface was actually proposed as a scale space operator. Yet these iterated projections cannot be consistent with MCM, because by definition they do not evolve degree two surfaces. Furthermore, the first step of MLS2 is always to compute a regression plane, which gives the estimate of the normal. We have just shown (Theorem 9) that this computation by itself is sufficient to get directly not only the normal, but also the surface motion by mean curvature.

Can iterated MLS2 give a better estimate of the curvature than the projection filter? Comparative experiments were performed on a randomly and uniformly sampled sphere with added gaussian noise. The point samples were filtered four times by T_r. By Theorem 9, each filtering step gives an estimate of the mean curvature. The same sampled sphere was filtered by MLS2, the surface mean curvature being computed as the mean curvature of the approximating surface at each point. This estimate is very exact, since the difference on a C^4 surface between a point and its MLS2 estimate can be proved to be $O(r^4)$. Both mean curvature estimates are compared by their mean and standard variations in the table of fig. 3.3. The result shows that when the noise level increases the planar projection yields a much more stable computation (see the fast decay of the standard variation for the curvature estimate). This experiment is also coherent with the MCM consistency theorem. Indeed, the planar projection yields a point set with (slowly) increasing curvatures (once the noise is removed, i.e., once the standard variation is stable).

Fig. 3.4 is another illustration in 1D of the same phenomena: a circle with radius 1 and added gaussian noise with variance 0.05 is denoised by iterated T_r and by an iterated MLS2 projection using the same neighborhood radius. In 1D, T_r becomes a simple line regression and the MLS2 surface a degree 2 polynomial curve. The simplest MLS2 method is used: it merely performs a weighted least squares polynomial regression on the local neighborhood. The neighbors weights are equal to $G(d)$ where G is a gaussian and d is the distance between the neighbor and the center point. The standard variation of G is equal to the neighborhood radius.

Noise	0.01	0.05	0.1
Plane 1	1.00/1.95	1.15/5.57	1.27/4.76
Plane 2	0.99/0.07	1.02/2.16	1.17/4.89
Plane 3	1.00/0.02	1.01/0.16	1.05/2.10
Plane 4	1.01/0.01	1.01/0.05	1.02/0.27
Plane 10	1.04/0.01	1.05/0.01	1.09/0.04
MLS 1	0.94/0.22	0.11/2.58	−0.42/2.99
MLS 2	1.01/0.13	1.02/0.49	0.62/1.36
MLS 3	1.01/0.10	1.02/0.36	1.06/0.68
MLS 4	1.00/0.08	1.02/0.30	1.05/2.19
MLS 10	1.00/0.04	1.01/0.14	1.02/0.74

Figure 3.3: Comparison of mean curvature estimates on a noisy sphere with radius 1 (mean/standard variations) given by iterated planar projection and iterated MLS2 regression. The curvature is evaluated at all points as the displacement along the normal induced by the planar projection (as stated in Thm 9) for the planar case, and by the explicit computation of the MLS surface mean curvature in the MLS2 case. The same radius is used for both iterations and both regressions.

3.4 First application: scale space raw data point orientation

Given an initial non oriented raw point cloud the surface orientation is a much needed information before meshing. The eigenvector of the least eigenvalue of the local covariance matrix is a classic approximation of the normal. We must then pick one of two possible orientations, and this choice must be globally coherent. The idea is to start by picking a random orientation for one point and to propagate it to the neighboring points. Now, sharp edges or a noisy surface could fool such a propagation. If, however, the surface is smooth enough, the propagation of the normal is safe. Thus the overall technique to orient the raw data set will be to smooth it by the scale space, to orient the smoothed surface, and to transport back this coherent orientation to the initial data points.

The first tool to realize this program is a simple propagation method for a point p whose neighborhood $\mathcal{B}_r$ contains some previously oriented points. The orientation is transmitted from one point to the next if their normal directions are similar,

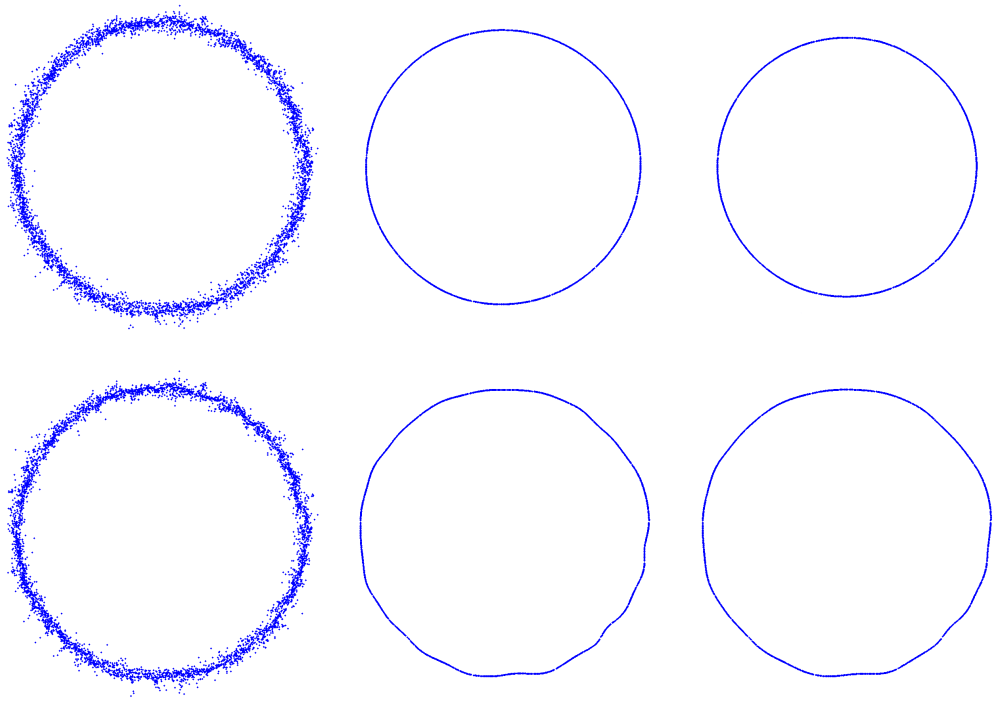


Figure 3.4: Denoising a noisy circle with $1, 50, 100$ iterations of T_r (top) and MLS2 (bottom). Even after 100 iterations the oscillations removed by T_r persist with MLS2. The sphere radius decreases with T_r, which is consistent with the mean curvature.

i.e., their angle is below a certain threshold (algorithm 1).

Algorithm 1: OrientateFromNeighbors(p,r,t)

Data: p an unoriented point, a threshold $0 < t < 1$, a radius r, the set $\mathcal{B}_r$ of p's neighbors within radius r
Result: true if the point was oriented, false otherwise

```
Compute p's normal direction n by local PCA;
n̄ ← normalized mean of already oriented neighbors' normals;
if (n̄ · n)² > t then
    if n̄ · n > 0 then
        n(p) = n;
    else
        n(p) = −n;
    Return true;
else
    Return false;
```

Algorithm 2: Scale space Orientation

Data: A point cloud $\mathcal{M}_S$, a radius r, an update parameter $\alpha > 1$

```
Iterate the projection filter T_r and keep track of each raw data point sample
(mean curvature motion);
Find a point p_0 in a flat area, pick its orientation and mark it as oriented.
Add its neighbors to the stack S;
while S is not empty or S does not become constant do
    Take p_0 the first point in S;
    if orientateFromNeighbors(p,r,t) then
        Mark the point as oriented and remove p from S;
    Add p's neighbors to S;
Add all remaining unoriented points to S;
while S is not empty and ♯S does not become constant do
    r = αr;
    for p in S do
        Perform orientateFromNeighbors(p,r,t);
```

The input parameters for the scale space orientation (algorithm 2) are the radius r and a threshold $0 \leq t \leq 1$. Steps from 8 to the end are necessary because adding neighbors of points to the stack might not be enough to cover the entire cloud due to sampling irregularities. Once this procedure is over, there might remain non oriented points. These points are usually isolated points, and we choose to ignore them. In all our experiments the number of remaining non oriented points was

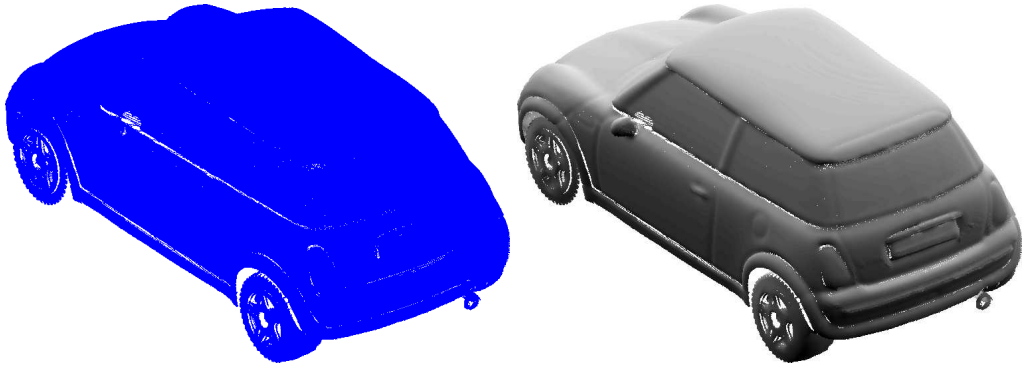


Figure 3.5: A raw point set (left) and its orientation (right). Points in the right figure are simply given a gray value depending on the scalar product of their normal and the lighting orientation

below 0.1%. At each step the radius is multiplied by an $\alpha > 1$ factor. In step 10, the radius r is changed. Thus all normals are not computed with the same radius. This is why we must reverse the scale space to come back to the original point cloud. At scale 0, the normal direction is recomputed by local PCA for all points and the chosen orientation is the one which has positive scalar product with the previous normals. This is an application of the scale space paradigm, where the information is computed at a coarse scale and propagated back to the finest scale.

A result of this orientation algorithm is shown on a car point set (Fig. 3.5).

3.5 Second application: scale space meshing

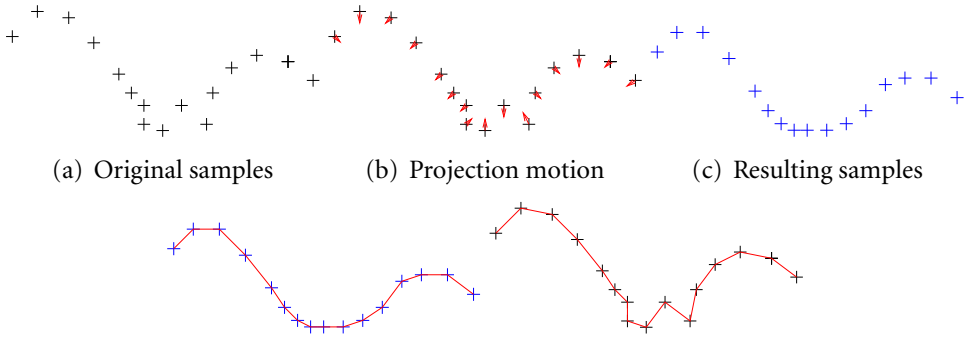
We now discuss how to build a mesh on the raw point cloud. Direct meshing is not possible because of the surface oscillation due to fine texture and noise (see Fig. 3.9(c)). The idea is again to perform meshing on the smoothed surface and to transport this mesh back on the original point cloud. An efficient triangulation technique, the ball pivoting method [BMR*99] is used in all experiments. The crucial faithfulness requirement is that the final vertices of the mesh must be an overwhelming majority of the raw data set points. This conservative requirement, incompatible with level set methods ([KBH06], [HDD*92], [LC87]) is described in Algorithm 3.

Fig. 3.6 illustrates why scale space meshing allows one to recover those details: standard meshing at a smooth scale is simply easy because details have been unfolded. It is then trivial to propagate back the vertices of the smooth mesh to their initial positions. This yields a direct triangulation of the original raw data set.

Parameters of scale space meshing The radius can be set automatically while com-

Algorithm 3: Scale space meshing

Data: A point set with computed normals
Result: A mesh of the original 3D data point set
1 Iterate (four times) the projection filter T_r and keep track of each raw data point sample: this is the forward mean curvature motion;
2 Mesh the smoothed samples;
3 Transport the mesh back to the original points (thus reverting the mean curvature motion).



(a) Original samples (b) Projection motion (c) Resulting samples

(d) Mesh of the resulting samples (e) Scale space Mesh

Figure 3.6: 2D example of the steps performed by the scale space meshing algorithm

puting the octree to sort the points. Indeed the root of the octree is the bounding box of all points. Let us call L_{max} the length of its largest side. Then, each cell represents a 3D cube with size $L_{max}/2^d$ where d is the depth of the cell. Counting the number of points in that cell gives an approximation of the number of neighbors of a point contained in this cell for a spherical neighborhood of radius $r_d = L_{max}/2^{d+1}$. Performing this approximation in all non empty cells at the same depth gives an approximation of the number of neighbors for spherical neighborhoods with radius r_d. The projection filter requires at least three neighbors per point to estimate a regression plane, but a robust estimate is experimentally attained with 30 neighbors. Of course, since the same radius is used for all points, it may occur that there are not enough neighbors to perform the plane regression. Those points must be eliminated, but in all the experiments less than 0.1% of points were removed this way. These points are mostly outliers, or isolated points in folds of the acquired object. Although their relative number is low, nonetheless this represents some thousands points that are eliminated.

Once the minimal number of neighborhood points has been fixed (and it has been fixed once and for all on all experiments to 30), the radius is also fixed and the

meshing scale space only depends on the number of scale space iterations. When setting the radius automatically as described above, it was found that four iterations were always enough to smooth the point cloud and build the mesh. Thus, the scale space is conceived as a very local motion securing a reliable tangent space. In all experiments the points barely moved (less than 40μ for the Tanagra point set). The scale space and the ball pivoting reconstruction use the same ball radius. The parameter of the Poisson reconstruction (namely the octree depth) was set to be the largest allowed by our computing equipment (namely a 8 $3Ghz$ processors computer with 48 Go RAM).

Transporting back the connectivity information (step 3) can in theory lead to a self intersecting mesh. Indeed, if two points lie too close to each other they may "switch position" in the scale space iterations, leading to a complicated surface topology. This problem can be solved by detecting all pairs of intersecting triangles. Then any remeshing algorithm can solve the problem by switching edges in quadrilaterals. However, this additional step was not implemented for two good reasons. First, the existence of a few intersecting triangles would be no serious visual inconvenience. Second, no such crossing was found in any of about twenty experiments on very large data point sets.

3.6 Comparative experiments on high resolution data

The algorithms were devised for highly accurate point clouds acquired by a laser scanner. A typical example of the scanned objects is a mould of a fourth century B.C. Tanagra figurine acquired at the Museum of Cycladic Arts, Athens (Fig. 3.7(a)). It is 22cm high and the point cloud contains $6 \cdot 10^6$ points.

Thanks to a very accurate calibration of the laser scanner device, the output is a well registered non oriented point cloud containing a negligible warp. Tests were also made on objects of the Stanford Fragment Urbis Romae database. In that case a registration of the raw sweeps is needed to have a point cloud representing the whole object. Preferring not to address the sweep registration problem in this chapter, we will use single sweeps for our meshing experiments and show that considerable texture information can be recovered from each sweep. Figs. 3.7 and 3.8 show the application of scale space meshing with a mesh rendering at fine and coarse scale. We can see on Figs. 3.7 and 3.8 that the surface texture is lost at a coarse scale, but completely and accurately recovered by scale space meshing. Comparing the back projected mesh to the result of a direct meshing of the initial samples (Fig. 3.9) shows that the scale space triangulation is much more precise. In fact, a direct meshing is not applicable. It creates, among other artifacts, many spurious triangles. T_r has been proposed as the simplest smoothing operation implementing the mean curvature motion. It may be objected that the surface could also be

directly approximated by the classic order 2 moving least square method (MLS2). The most objective way to compare T_r and $MLS2$ was to implement them with exactly the same neighborhood radius. Fig. 3.9(f) shows the comparative result on one of the finest details of the Tanagra data set. The results are similar in terms of detail quality, yet the computation time was doubled, and we have seen (Fig. 3.3) that MLS2 does not deliver a scale space and keeps the smoothed out noise. Fig. 3.13 shows a comparison between the reconstruction obtained by the VRIP reconstruction method (see [CL96]) and scale space meshing. The scale space method produces a significantly more precise mesh, as can be seen on the close up of Fig 3.14.

Fig. 3.10 shows the scale space reconstruction of one scan of a fine scale object (i.e. the mesh back-projected at all scales).Fig 3.11 compares the mesh reconstruction by several meshing algorithm with scale space meshing. The experiment clearly rules out both Ball Pivoting algorithm and Poisson Reconstruction. Two MLS methods are also tested. APSS ([GG07]) and RIMLS ([OGG09]). APSS builds an implicit function by evaluating the distance between each evaluation point and an algebraic spherical fit of the surface. Though the method is not explicitely deviced for meshing, the isosurface can be extracted using the marching cubes. RIMLS is another modification of the standard MLS procedure. It is based on minimizing an objective function that gives less weight to spatial and normal outliers (i.e., sparse points and features). Here, marching cubes are explicitely mentioned for extracting the surface. For both methods, the resolution depends on the marching cubes grid resolution: it was set so that increasing it would not change much the visual aspect. Though both methods are visually close to scale space meshing, our method is much simpler and does not use an isosurface extraction at all. It is also the only method which preserves input samples and does not add additional vertices (both APSS and RIMLS use more than twice the number of input samples). Another problem is that the isosurface extraction by marching cubes introduces strong artefacts which are avoided by scale space meshing (Fig. 3.12).

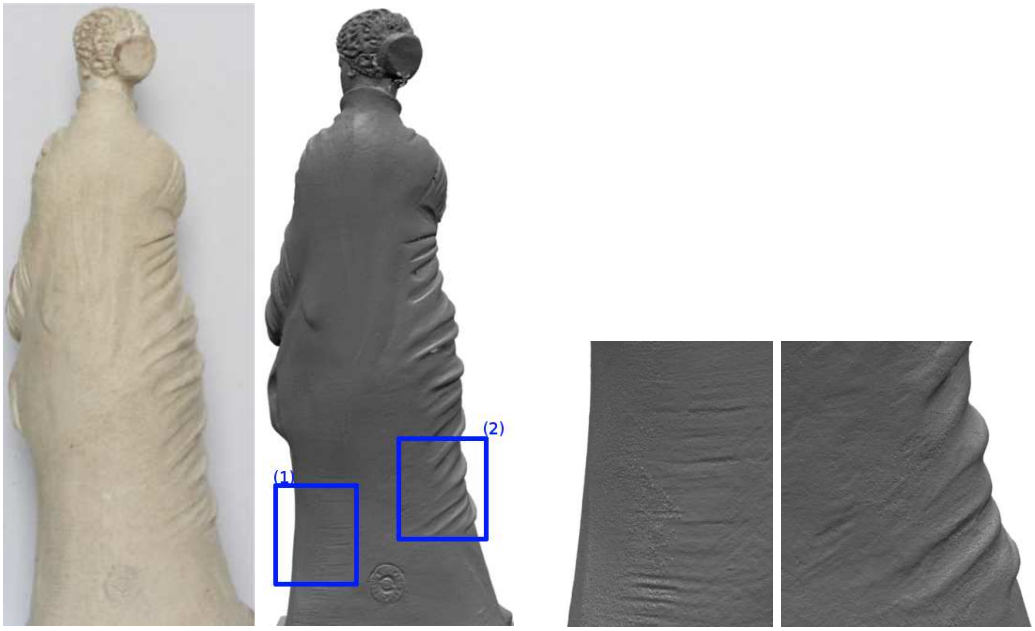
Fig. 3.7 displays the many acquisition holes at the bottom of the Tanagra figurine, in the folds of the tunic or near the right foot. By the scale space meshing these holes are not filled in and can be detected. Since the ball pivoting algorithm is used for triangulation, no triangle larger than a given threshold has been created. Indeed, to form a triangle, three points must lie on a sphere of given radius r. Thus, low density areas are considered as holes.

Fig. 3.9 illustrates the loss of detail with level sets methods. Level set methods create a smoothed zero level surface of the signed distance to the raw data set point. They do not contain the raw data set points and lose track of them. Fig. 3.15 shows that not only these methods, but even direct meshing methods can miss small details.

The quantitative performance of each algorithm can be evaluated by meshing

(a) Initial object (b) Coarse scale mesh (c) Scale space mesh

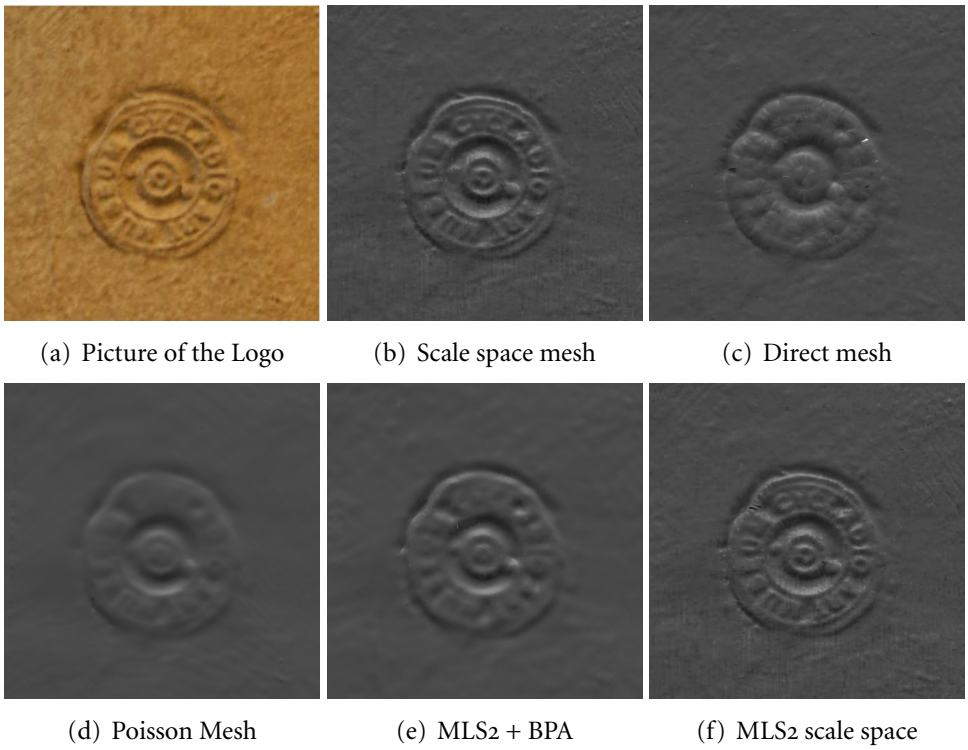
Figure 3.7: Multi-resolution mesh reconstruction of the Tanagra point set (22 cm high) illustrating the recovery of fine texture. All back propagated textures are present on the original



(a) Picture of the back side (b) Detail selection

Figure 3.8: Close ups on the details (1) (left) and (2) (right) selected on fig. 3.8(b)

simple shapes. Test point sets were built by sampling perfect geometric shapes (for example a sinusoidal surface). The root mean square distance of the triangle barycenters of the mesh to the real surface were compared for each meshing method. This distance is computed by the Newton-Raphson method. The first surface "Wave 1" has equation $z = 0.2\cos(5x)$, "Wave 1" has equation $z = 0.2\cos(5x)*\cos(5y)$, the third surface is a regularly sampled sphere and the last one is a sum of two close and narrow Gaussians $z = -\exp -\frac{(x-0.1)^2}{0.01} - \exp -\frac{(x+0.1)^2}{0.01}$. The RMSE results are shown in the table of fig. 3.16. It is obvious from these results that the Poisson reconstruction or any level set method cannot be applied to recover a surface with very thin details. On shapes containing no sharp edges, direct BPA and scale space meshing perform comparably. On the thin structure created by adding two very close Gaussians, the loss of precision of BPA is clear. This phenomenon is similar to the one observable in Fig. 3.9(c) where BPA does not recover thin details.



(a) Picture of the Logo (b) Scale space mesh (c) Direct mesh

(d) Poisson Mesh (e) MLS2 + BPA (f) MLS2 scale space

Figure 3.9: Comparison between several meshing methods. The width of this logo is approximately 1cm. The direct mesh (3.9(c)) creates many spurious triangles. The Poisson reconstruction [KBH06] clearly smoothes out all details (3.9(d)). Filtering the logo by order two MLS and meshing the points by the ball pivoting algorithm (3.9(e)) also creates a smooth mesh. Compare the details in 3.9(b) and 3.9(c). Fig 3.9(f) shows the result of applying the same scale space strategy with the projection on the order 2 MLS surface instead of the regression plane projection operator. The result is alike in detail quality but the computation time is double. See figs 3.6-3.15 for an explanation of this difference.

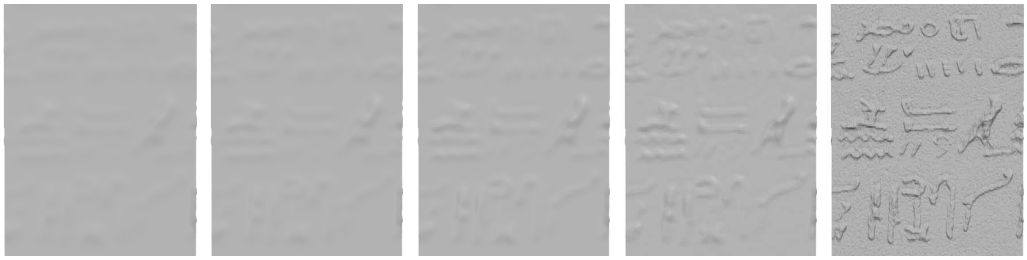
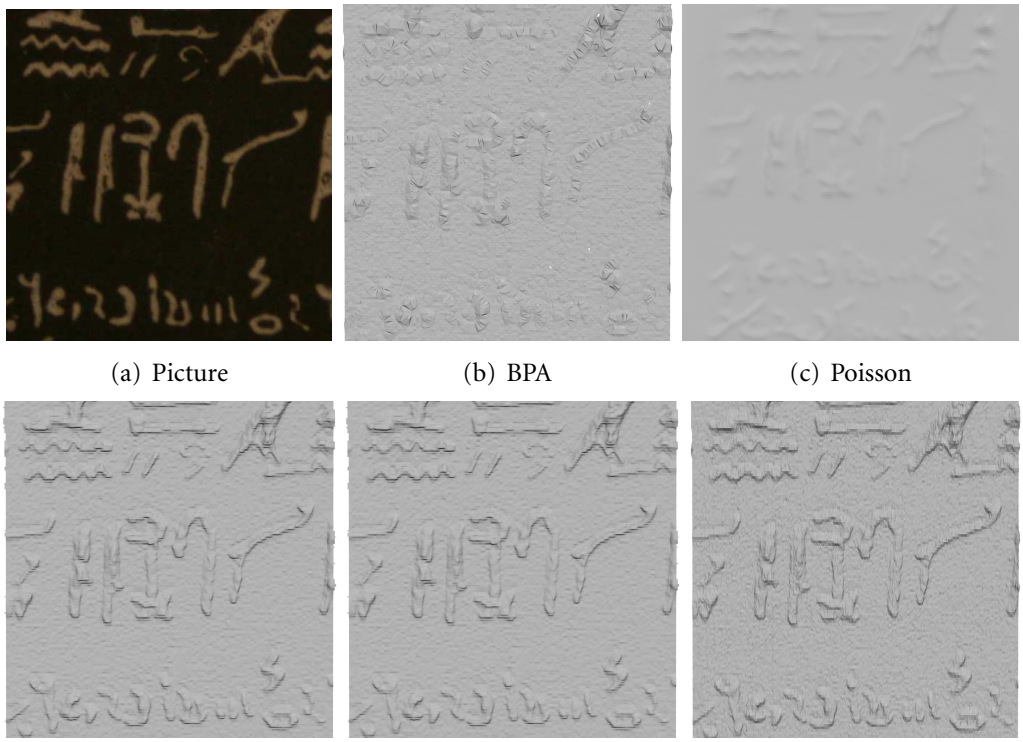


Figure 3.10: Back-projecting the mesh of a single scan of a fine-scale object (engravings are around $0.1mm$ deep). From left to right: mesh built after 4 scale space iterations; back-projection of the mesh to levels $3, 2, 1$; back-projected mesh (final mesh)

3.7 Complexity analysis and computation time measures

One scale-space projection requires the following operations: look for neighbors within radius r, build their covariance matrix and their centroid, perform PCA of this 3×3 covariance matrix. Therefore, once the neighbors are found, they are sequentially scanned in order to build the covariance matrix and the centroid. This yields 6 multiplications and additions per point for the covariance matrix update and 3 additions per point for the centroid update. The PCA complexity does not depend on the number of neighbors: it requires 9 operations. Knowing the least eigenvector, the projection is only 12 operations. There is one list scan (9 operations per processed point) and 21 operations once the covariance and centroid are built. Assuming we have 30 neighbors, this yields a total of 200 operations per point. Finally finding the neighbors in an octree structure is $O(logN)$ (average) and one scale space iteration therefore is $O(N(logN + 200))$ operations, where N is the total number of points in the point cloud.

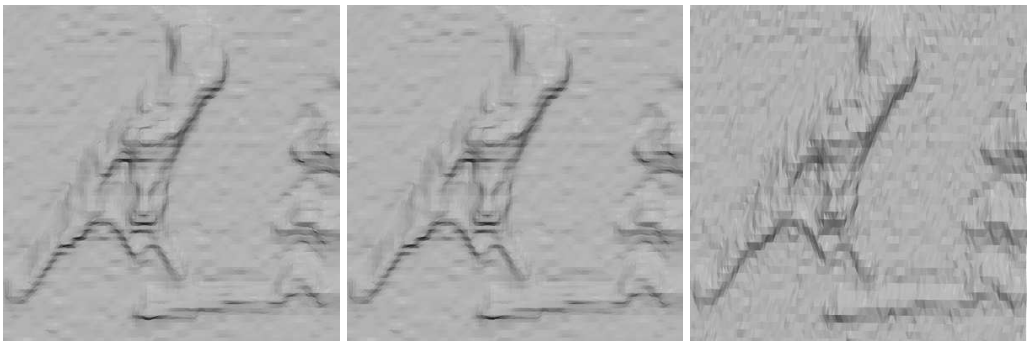
The computation time needed for meshing the Tanagra point set with six millions points was as follows: Sorting the points in the octree takes $1.2s$. The scale space iterations requires 3 min. This leads to a total computation time of $19min$ for the scale space point cloud orientation and of $27min$ for the whole scale space meshing on an 8 $3Ghz$ processors computer with 48 Go RAM. The maximum memory usage was less than $2Go$. These figures should be compared with the time required for directly meshing the oriented point set by the ball pivoting method without any scale space iterations, which took $25min$. Therefore, only a two additional minutes were used to get a much more faithful mesh.



(a) Picture (b) BPA (c) Poisson

(d) RIMLS (e) APSS (f) Scale space meshing

Figure 3.11: Comparison of the Rosette reconstruction (Picture (a)) using Ball Pivoting Algorithm (b), Poisson Reconstruction (c), RIMLS (d), APSS (e), and scale space meshing (f). APSS and RIMLS yield results that are really close to ours, yet both methods need an isosurface extraction done with the marching cubes, which creates strong artefacts (see a closeup Fig 3.12). Besides, RIMLS and APSS meshes contain around 268500 vertices whereas the scale space mesh contains 132203 vertices. Notice also that APSS and RIMLS introduce some denoising (visible especially in the nearly flat parts). Scale space meshing is the only method that preserves exactly the input data.



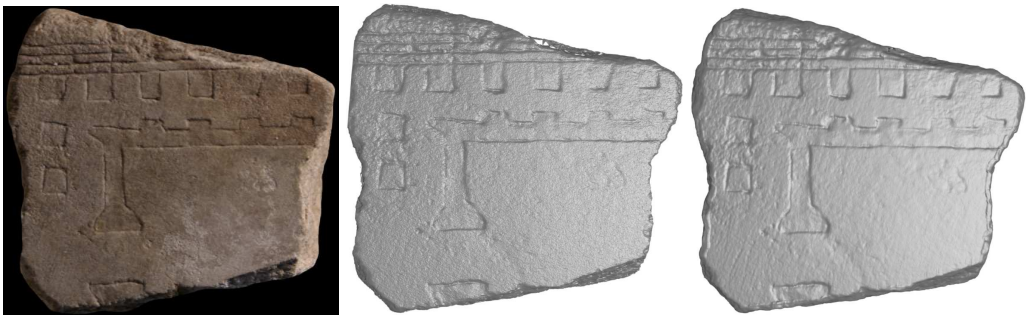
(a) RIMLS (b) APSS (c) Scale space meshing

Figure 3.12: Detail of the mesh built using (from left to right) RIMLS, APSS and scale space meshing. Notice the horizontal artefacts created by the isosurface extraction for both APSS and RIMLS methods.

3.8 Conclusion

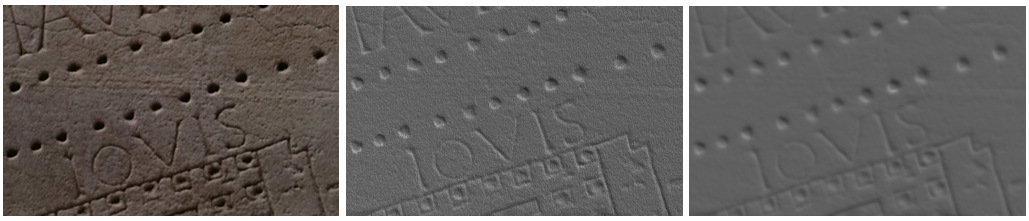
The increasing accuracy of 3D triangulation scanners requires an effort to reconsider the whole rendering chain, and to obtain high quality visualization, actually better than what is obtained by photography. The present chapter has proposed a strategy to mesh the raw original surface, therefore ensuring a faithful rendering. Future work will test a closed scanning loop with our experimental scanner to scan the detected holes.

Acknowledgements The authors thank the Stanford Digital Forma Urbis Romae Project (http://formaurbis.stanford.edu) and professor Marc Levoy for the fragments data used in figures 3.13, 3.14 a joint property of the Sovraintendenza of Rome and of Stanford University.



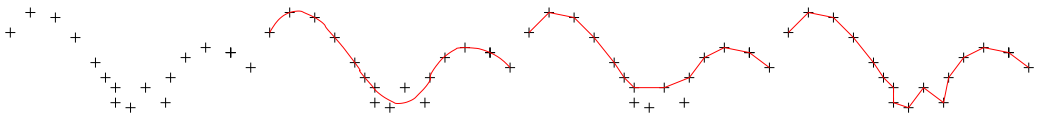
(a) Picture of the fragment (b) Scale space mesh (c) Reconstructed Mesh available on the FUR website

Figure 3.13: Comparison on a piece of the Fragment Urbis Romae (FUR) database. Texture and details are better recovered on the back-propagated mesh (middle). Compare with the VRIP reconstruction method available on the FUR website (right)



(a) Original Fragment (b) Scale space mesh (c) Level Set mesh

Figure 3.14: Closeup of a piece of the (FUR) database reconstructed by Scale Space Meshing and by the Poisson Reconstruction Method.



(a) Original samples (b) Level set method (c) Direct Meshing (d) Scale space meshing

Figure 3.15: Comparison of three meshing methods

Method	Wave 1	Wave 2	sphere	sharp
Scale space	0.19	0.28	0.04	0.04
BPA	0.18	0.24	0.04	1.2
Poisson	1.5	43	0.24	4

Figure 3.16: Quantitative comparison of three meshing methods, scale space meshing, ball pivoting, and the Poisson reconstruction: RMSE of the distance from the triangle barycenters to the real surface. All results are multiplied by 10^3 for readability

Chapter 4

High Fidelity Scan Merging

Contents

Abstract: For each scanned object 3D triangulation laser scanners deliver multiple sweeps corresponding to multiple laser motions and orientations. The problem of aligning these scans has been well solved by using rigid and, more recently, non-rigid transformations. Nevertheless, there are always residual local offsets between scans which forbid a direct merging of the scans, and force to some preliminary smoothing. Indeed, the tiling and aliasing effects due to the tiniest normal displacements of the scans can be dramatic. This chapter proposes a general method to tackle this problem. The algorithm decomposes each scan into its high and low frequency components and fuses the low frequencies while keeping intact the high frequency content. It produces a mesh with the highest attainable resolution, having for vertices all raw data points of all scans. This exhaustive fusion of scans maintains the finest texture details. The method is illustrated on several high resolution scans of archeological objects.

4.1 Introduction

Recent high precision triangulation laser scanners can scan surfaces of medium size objects with a precision of less than 10μ. Yet, although each scan has a very high precision, this precision can be lost again when merging multiple scans and meshing them together. This loss of precision entails a loss of visible texture, which explains the smooth and glassy aspect of most rendered scanned objects. On the other hand the merging of the multiple scans (often called *super-resolution*) is absolutely necessary. A patch of the object may well be acquired tens and even hundreds of times on well exposed parts. Indeed, many sweeps with varying trajectories are necessary to acquire the less exposed parts of the object. The main goal of the merging considered here is not to gain more detail and texture or to denoise the data point cloud by super-resolution: recent triangulation scanners yield scan sweeps with excellent quality. Unfortunately this quality is at risk of being damaged by the merging procedure itself. Thus, more trivially, the goal is to secure that the texture of each scan is not lost again due to slight matching errors which force a smoothing before a joint meshing. Fig. 4.1 illustrates the problem. With two overlapping shifted scan grids, as seen in (a), the aliasing risk is high. Meshing each scan separately yields two almost identical surfaces and textures (b, c). Nevertheless, a joint meshing (d) provokes strong tiling and aliasing effects, due to very small local offsets between both scans, in spite of the fact that they have been globally well registered. The challenge is therefore to merge both scans in such a way that the rendering quality does not decrease. The numerical problem is made more acute by two facts. First, not just two, but up to hundred scans may overlap in some region. Second, scans

boundaries appear everywhere, as illustrated in fig. 4.2 and make the fusion near these boundaries still more problematic.

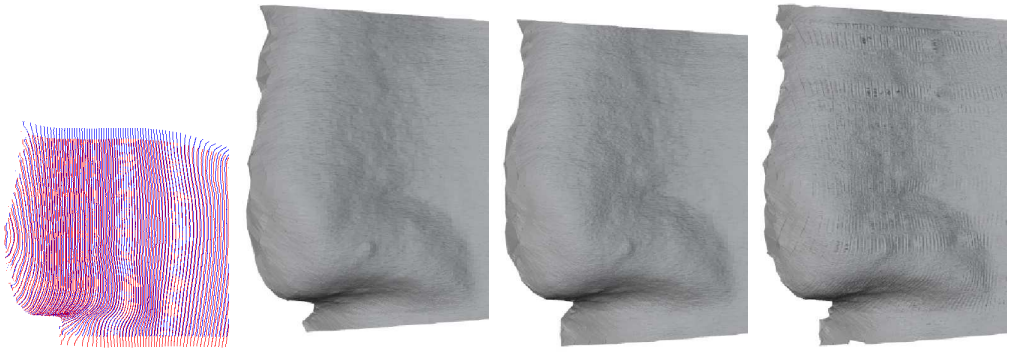
Each point of each scan has three-dimensional coordinates given either in a global coordinate system if the acquisition device is calibrated, or in a local coordinate system if the device is not calibrated. In the case of non-calibrated devices, the scans must be registered in a common coordinate system, and the registration problem becomes a rigid transform estimation. This problem has been widely investigated and has found efficient solutions [BM92] [RL01]. Yet if the scans had some internal local warping (which is usually the case), the rigid transform framework is not sufficient. A whole theory of non-rigid scan registration has therefore been developed [BR04],[BR07]. If the acquisition device is well calibrated the delivered scans are well registered, up to a given precision. Yet, as we already mentioned, a tiny residual mismatch can provoke strong artifacts similar to aliasing patterns (see fig. 4.1) and forbids a direct meshing of the union of all data point clouds. This problem is generally solved by applying a method which meshes an implicit zero level set of a distance function to the raw points. The distance function is approximated by its Fourier coefficients [Kaz05] or by radial basis functions [KBH06]. The problem is that these methods result in a serious loss of accuracy when the final result is compared to each scan separately.

This chapter experiments on sets of scans of an object that have been either previously optimally registered by rigid or non-rigid methods, or registered through a high precision calibration of the acquisition tool. To demonstrate that no texture content will be lost, the goal is to mesh the entire point cloud. This means that all raw acquired points of all scans will ideally be vertices of the mesh. This requirement guarantees a complete preservation of all the acquired information, including noise and fine textures. Of course such a mesh is not numerically economic, but it is necessary for two goals: first to get high quality rendering of complex shapes such as archeological objects, and second to precisely explore all remanent artifacts such as the holes, inherent in any scanning process. For scanning control purposes, it is anyway quite rewarding to be able to *see exactly* what has been scanned.

4.2 Previous Work

4.2.1 Rigid Scan registration

When the scans contain no warp, the registration problem sums up to estimating a rigid transform between scan coordinate systems by minimizing the distance between the reference scan and the transformed scan. The estimation of the transform in the quaternion form was proposed in [Hor87]. The Iterative Closest Point (ICP) Registration procedure introduced in [BM92] and [CM92] matches a point



(a) Overlapping scans (b) Mesh of scan 1 (c) Mesh of scan 2 (d) Mesh of scans 1&2

Figure 4.1: Example of two overlapping scans (a), points of each scan are first meshed ((b)-(c)) separately. The result can be compared to the meshing of points of both scans together (d)

Figure 4.2: Example of overlapping scans. This head is such a complex structure that not less than 35 scans were acquired to fill in most holes.

from one scan to the closest point of the other scan and computes the transform based on these matchings. ICP was very successful and many variants were introduced ([RL01],[BS97]). A study of the optimal sampling for the ICP algorithm was proposed in [GIRL03].

Since ICP converges to a local minimum, the initial scan position must be close to optimal. Thus, some robust initial matches are needed to initialize the algorithm. This robust match search has also been investigated. The common idea is to find features easily identifiable on the scans (usually linked with highly curved points) and to match them. Spin Images, introduced in [JH99], represent potential feature points by images. The spin image associated with point p is the function that maps a 3D point to (α, β), where α is the distance to the tangent plane and β the distance to the line parallel to $\mathbf{n}$ and passing through p. A linear correlation coefficient is used as a similarity coefficient between images to find the correspondences. In [SA01] another feature carrier is introduced. It is based on computing geodesic circles around a point and projecting them onto the tangent plane. This gives a $1D$ 2π-periodic function parameterized by the angle. Feature matching is done by sampling the contours and computing a similarity measure between the contours. Other popular descriptors were described in [YF99], [VSR01], [ZH99] or [KFR03].

Another approach was proposed in [AMCO08]: all coplanar 4-point sets that are approximately congruent are extracted on both shapes and matched using the fact that the distance ratios relatively to the intersection point are invariant to rigid motions.

Rigid scan registration methods assume that the scans must fit perfectly using only rigid transforms. Yet, if the scans have some warping, the method does not apply. Another problem is that the registration error might cumulate when registering multiple scans (see [BR07] for examples of bad registrations). These considerations led to modeling the registration transform by a non rigid transform as will be shown in the next subsection.

4.2.2 Non-rigid scan registration

The method for non rigid registration using thin plate splines [BR04] first applies a hierarchical ICP to find good features: the source mesh is iteratively divided through the middle of its longest axis and each half is realigned separately. This yields good feature correspondences at the cost of substantial discontinuities in the source mesh. These point matches are then used to compute the thin plate spline that best approximates all pairs of points. The spline being continuous, any discontinuity introduced in the scan splitting process is removed by the spline approximation. Once the scans are registered they must be merged. This is done using the VRIP method [CL96], which will be described below.

An extension of this method was introduced in [BR07]: the same thin plate

spline non rigid deformation model is applied using an improved earlier point matching: features are found and matched in a process which rejects outliers. Other methods include [CR03] where the non rigid registration is done using as input soft assignment between point pairs. This yields a functional minimization comprising a term of soft assignment.

4.2.3 Super-resolution from several scans

Recently, the problem of achieving super-resolution from multiple scans has been raised, mostly for improving range image resolutions ([KMA06]) where various low resolution scans with the same depth direction are acquired and registered by ICP. Depth values at each point of a high resolution grid are then interpolated with depth values of points falling in the neighboring cells. [AKSA09] used rotating scans around the depth axis to build for each orientation high resolution range images. These high resolutions range images were registered and combined by considering the image gradient and the angle between the baseline and the image gradient to weight each point. Other methods building hybrid scanners to achieve high resolution include [NRDR05] where positions and normals are acquired and used to improve the resolution.

Once scans have been computed and registered, a mesh must be built to allow a fast visualization of the surface model. The next section reviews methods to build the mesh.

4.2.4 Meshing

Meshing methods can be divided into two categories: methods that approximate the point cloud and methods that mesh directly the point set. Approximating methods usually build a function defined on $\mathbb{R}^3$ whose 0-level set is the shape surface. A mesh is then built on the 0 level-set by the marching cubes algorithm [LC87]. These methods include [KBH06],[Kaz05], among others. A very interesting variant of these methods is the VRIP algorithm ([CL96]). VRIP considers an implicit function taking into account not only point positions but also their reliability. Nonetheless, two drawbacks common to the mentioned methods are the automatic filling in of holes, and the implicit low pass filtering performed by the level set method. These methods usually compute the distance to the surface as an average of the signed distance of the point to its k-nearest neighbors. Thus initial points are discarded and *de facto* replaced by local averages. This removes noise in the cloud, but also loses fine details and textures.

Direct meshing methods include [ABK98] or [AB98]. We shall use the incremental ball-pivoting method [BMR*99], which is fast and does not fill holes. The method is based on pivoting a ball of fixed radius r around edges. Three points

are triangulated if they lie on a ball with radius r and empty interior. The ball is then pivoted around all three edges of the triangle, until it meets a point and has still empty interior. If no such point is met then the edge is the boundary of a hole. The parameter r is a bound for the creation of triangles: no triangle edge can have length above $2r$. Thus low density areas are considered as holes.

In short, the dilemma is as follows: the approximating methods are not sensitive to a slight registration error, but can loose detail and texture. On the contrary, aliasing-like artifacts become visible when a direct meshing method attempts to keep all raw points of several scans. The best choice is to preserve the raw points and to apply a direct meshing method. But this requires eliminating all traces of inaccurate registration.

4.3 Scan Merging

4.3.1 Principle

The general idea behind the scan merging method experimented here is to preserve point positions in non overlapping areas, and to make a fusion of the scans on overlapping regions while keeping all raw points. The fusion involves a smooth-base/height-function decomposition for each scan. The decomposition of a surface as the sum of a smooth base and of a height function was proposed for a different purpose in [KST09], and [ZTS09], where the height function was used to segment the mesh and extract features as contours of the height function. The underlying idea is that a surface $\mathcal{S}$ can be decomposed into a smooth base B and a height function h, so that:

$$\mathcal{S} = B + h$$

B can be seen as the low frequency surface and h can be seen as the high frequency term. Given several surfaces $\mathcal{S}_1 = B_1 + h_1, S_2 = B_2 + h_2, \cdots, \mathcal{S}_N = B_N + h_N$, the idea is to fuse the bases, but to keep exactly the h_i terms, thus preserving all fine details. In other terms, a common basis B for all surfaces must be found, the high frequencies of all scans adopting this common basis thereafter. This strategy is comparable to the one used for morphing applications in [PKG06]. In a way, the idea is similar to [SCOT03], where the high frequency error due to quantization was transformed into a low frequency error much less noticable.

The data merging using a high/low frequency decomposition has long been a classic method in image processing [BA83]. This article introduced the idea of separating each image into various frequency bands by a Gaussian pyramid. The low frequency bands were merged separately to obtain a smooth blending of different images. The method has been successfully used to create panoramas from multiple images [BL07] and texture 3D models [Bau02]. Two major differences are

that in [BA83] all frequency bands are merged, whereas the method proposed here only merges the low frequencies while keeping the high frequencies intact. Another important difference is the usage of a nonlinear heat equation instead of a linear frequency decomposition.

The next section addresses the robust decomposition of a surface into a base and a height.

4.3.2 Low/High frequency surface decomposition

Since the pioneering article [Tau95b] it is known that mesh high frequencies are removed by the application of the intrinsic heat equation $\frac{\partial p}{\partial t} = \Delta p$. Yet, our scanned surfaces are given as point clouds and not as meshes. A numerical scheme of the heat equation for raw point clouds must be used. This question has been addressed in [BSW09] and [PKG06]. We shall use the simple implementation of the intrinsic heat equation proposed in [DMMSL09]: this chapter proves that the intrinsic heat equation can be implemented by iterating a projection of each point onto the local cloud regression plane. Consider the projection operator T_r that projects each point p onto the regression plane of the neighbors of p enclosed in a ball of radius r. Then it can be proven that this motion is tangent to the intrinsic heat equation. The iteration of T_r yields a scale space (a representation of the shape at various smoothing scales). In all experiments r is set so that the ball $\mathcal{B}_r$ centered at p contains about 30 neighbors at almost all points, and the number of scale space iterations n is set to 4. The first parameter (30) is fixed so that a reliable regression plane is always computed. The second parameter, namely the number of iterations 4, is chosen to guarantee a smooth enough basis in all cases. It can be increased without damage. When iterating the projection operator with an initial surface $\mathcal{S}_0$, the surface $\mathcal{S}_t$ is iteratively smoothed. To each point p_t of $\mathcal{S}_t$ corresponds a point p_0 of $\mathcal{S}_0$, and the height function can be taken to be the vector $h(p_t) = p_t - p_0$.

An alternative definition for the height would be the scalar function $h(p_t) = (p_t - p_0) \cdot \mathbf{n}$, where $\mathbf{n}$ is the normal to $\mathcal{S}_t$ at p_t. Yet, the results with both height variants being fairly identical, the simplest definition was kept: it separates each data point into a smooth base point and a high frequency vector.

4.3.3 Finding a common smooth basis for all surfaces

Choosing a common basis for all scans is the next question. A natural constraint on the method is to keep fixed the points belonging to regions where only one scan is available. Finding the common basis then becomes straightforward: It is enough to apply the same number n of iterations of T_r with the same parameter r to all the sets after they have been put together. This global filtering assumes that the high

Figure 4.3: The unmerged head with aliasing artifacts (left), its smooth base (middle) and the merged result (right)

frequency term of the set $\mathcal{S} = \cup_i \mathcal{S}_i$ contains the registration error: when filtering $\mathcal{S}$ the registration error is filtered away (see fig. 4.3).

4.3.4 Algorithm

The method is summarized in Algorithm 4. The algorithm is based on two applications of the intrinsic heat equation scheme (here the iterated projection on the regression plane) with the same parameters and the same number of iterations. All registered scans are given in the same global coordinate system. The first application (Line 2) is done on the separate scans yielding the intrinsic high frequencies of each scan. The second application (Line 6) is done on all scans together. When filtering all scans together (lines 5 and 6) the registration error is suppressed and we get a common low scale registration or basis, the set of points $b(p)$. Adding back to them the high frequency component $\overrightarrow{b_i(p)p}$ restores all details from all scans.

An important feature of the method is that each region $\mathcal{A}$ of the shape that has been acquired by one scan only is not altered. Indeed, inside such a region, applying the separate scale space or the common scale space is strictly equivalent, since there is only one scan in the neighborhood of the points of $\mathcal{A}$. Then the point is first filtered to $b_i(p) = b(p)$, and therefore moved back to its original position p at Line 8. So in areas with only one scan, point positions are not changed. The only effect of the algorithm is the merging of overlapping scans.

Algorithm 4: Scale Space Merging

Data: N point sets (scans) $(\mathcal{S}_i)_{i=1\cdots N}$, a number of projection filter iterations n and a radius r

Result: The set of merged scans: $\mathcal{Q}$

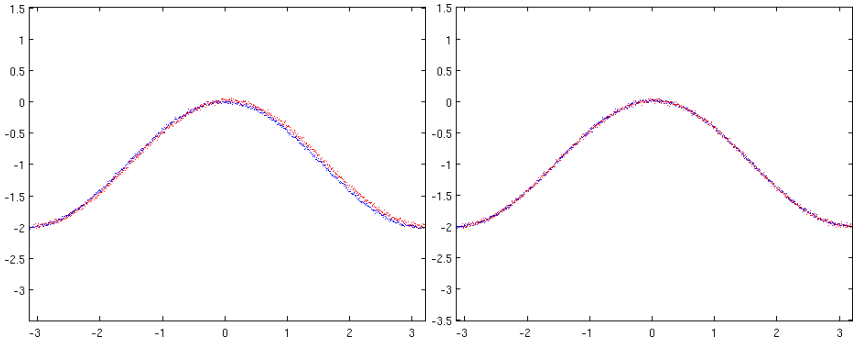
1 **for** $i = 1 \cdots N$ **do**
2 Apply n steps of the projection filter T_r to the set $\mathcal{S}_i$;
3 Store for each point $p \in \mathcal{S}_i$ with corresponding filtered point $b_i(p)$ the high frequency vector $\vec{\delta}(p) = \overrightarrow{b_i(p)p}$;
4 $\mathcal{S} \leftarrow \cup_{i=1}^{N} \mathcal{S}_i$;
5 Apply for each $p \in \mathcal{S}$ n steps of the projection filter T_r, yielding a point $b(p)$;
6 **for** $p \in \mathcal{S}$ **do**
7 $Q = b(p) + \vec{\delta}(p)$;
8 Add Q to $\mathcal{Q}$;
9 Return $\mathcal{Q}$;

RMSE	Both lines	Line A	Line B
Before Merging	X	$9.95e-04$	$9.76e-04$
After Merging	$9.85e-04$	$9.94e-04$	$9.75e-04$

Figure 4.4: Noise estimates on each separate scan A and B before and after their merging.

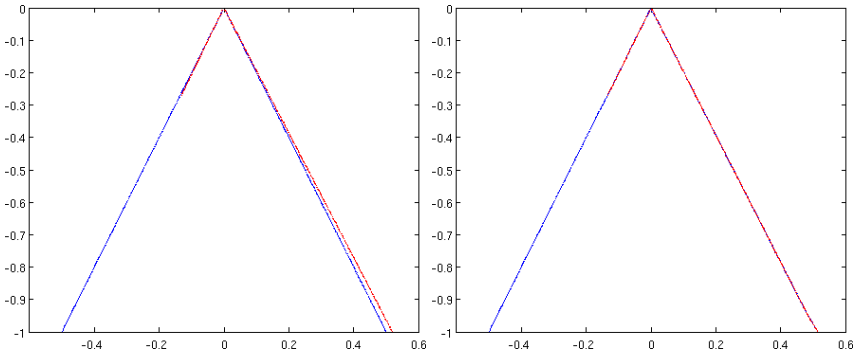
4.3.5 One-dimensional study

It is easy to illustrate the method in 1-D on simple $1D$ shapes. Our goal was to check that the proposed method superimposes two simulated scans without any smoothing effect. To do so, two noisy straight lines A and B were synthesized from the same model and then merged by the algorithm. The noise of each set A, B, $A \cup B$ was estimated as the root mean square error to their regression lines before and after merging. The results in Tab. 4.4 show that the merging did not cause any denoising. Indeed, the RMSE does not decrease by the merging procedure. Figs 4.5 and 4.6 show other 1D examples of the merging procedure where the bases are actually slightly different, in accordance with the real situation encountered on real scans.



(a) Without Merging (b) With Merging

Figure 4.5: Two noisy sine functions before and after merging.



(a) Without Merging (b) With Merging

Figure 4.6: Two noisy edges before and after merging.

4.4 Implementation and Results

The inputs of the merging algorithm are the outputs of our laser triangulation scanner. This device being accurately calibrated, the scans are in principle already registered so that no extra software registration is needed. Nevertheless, the ICP algorithm is applied to see if it can remove the aliasing and tiling artifacts. In this case ICP failes: the positions computed by ICP oscillate around the input scan positions, and the resulting meshes are no better. The registration process was implemented on a 1.5 Ghz processor with $48GB$ RAM. An octree structure was first built to allow for fast access to the neighbors of each given point. Table 4.7 gives the computation times for various shapes of various sizes with varying numbers of scans. Notice the high number of scans necessary to get a good covering of the object. It entails that several dozens of scans have to be merged on the more exposed parts.

Point set	points	scans	Time(s)	height
Dancer	$5,524,627$	94	321	$17cm$
Mime	$8,611,522$	102	140	$11cm$
Greek Mask	$8,961,736$	78	106	$12cm$
Nefertiti	$15,554,528$	115	819	$18cm$
Tanagra	$17,496,999$	160	1258	$22cm$
Rosetta	$36,201,537$	32	$45min$	$30cm$

Figure 4.7: Computation time for the proposed merging. It is significantly faster than the scanning time itself

Figs 4.8, 4.9 and 4.10 present the results on these data. For all point sets, two different renderings are displayed: the first one is a ball pivoting [BMR*99] meshing of all raw scan points without any merging. The scans were preregistered by the calibrated acquisition device and no software post-registration was needed. The second rendering is again a ball-pivoting meshing, but applied to the merged point set. The rendering was made using the POV-RAY ray-tracer. The conclusion is common to all experiments: even if the scans are actually very accurately registered, the tiny warps of the grids always create some aliasing visible as grid or tiling effects. After the merging procedure (which only slightly affects the low frequencies), these undesirable effects disappear almost completely. In the procedure more than 99.9% of the raw points were kept. Thus, the final result indeed is highly faithful to the raw scan. Yet a careful attention shows some remains of aliasing (Fig. 4.9, last column). The area of these is actually small, being inferior to the area of the holes. They could easily be removed by a selective local smoothing. Some of the bigger pieces, like Nefertiti, show no defect at all.

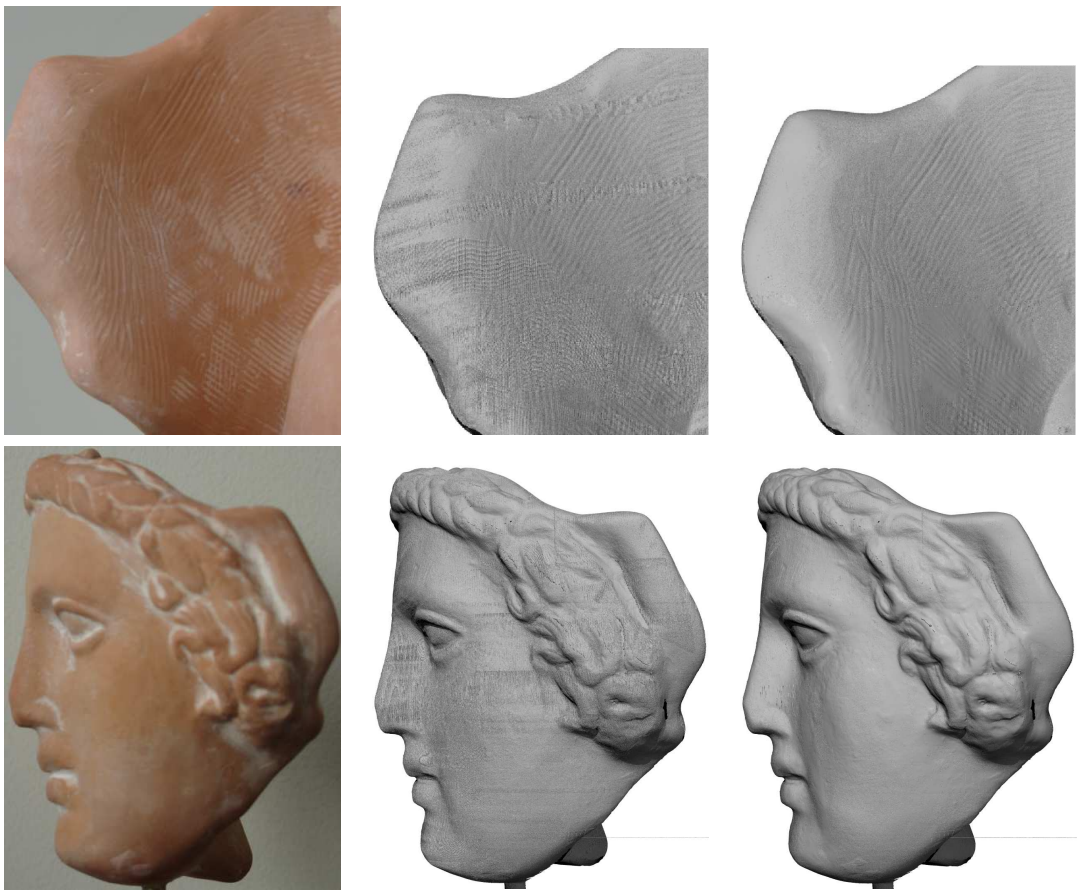


Figure 4.8: Merging of the mask scans seen from the back side (top row) and left side (bottom row). Left: picture, middle: without merging, right: with merging

Figure 4.9: Merging of the Dancer With Crotales. From left to right: picture, without merging, with merging, an example of merging failure taken from the back of the object (top: unmerged, bottom merged)

Figure 4.10: Merging of the Nefertiti (1st: picture, 2nd,4th: without merging, 3rd,5th: with merging)

Figure 4.11: Comparison of a rendering of a single scan (ground truth) and the merging of all scans that overlap in the same region (Left: ground truth, middle: joint mesh of all scans without merging, right: joint mesh with merging).

Comparison To better judge the texture preservation, the rendering of a scan alone (ground truth) was computed and compared to the rendering of all scans in the same region on Fig 4.11. This shows that the visual information loss after scale space merging is very low compared to the one due to a simple joint meshing.

It is crucial to compare the raw merging method results with results obtained by the level set reconstruction method of the unmerged scans point set. The result of the level set method applied to the Tanagra head (fig. 4.12 b), obviously introduces an important smoothing and loses texture in comparison to the merging result (fig. 4.12, a). But even with that smoothing the result still keeps several artifact lines due to the scan offsets: these offsets become visible at the scans boundaries. See the nearly straight long lines on the surface, mostly vertical and horizontal. It can also be asked if an efficient denoising method could actually restore the raw set. Fig. 4.12-c, shows the result of the application of the bilateral filter [FDCO03] to the union of the scans. This iterated filtering was applied up to the point where aliasing artifacts were no more visible. Clearly, this entails a much too strong smoothing of detail and texture.

The scan merging is a very local method which is therefore computationally efficient (see Tab. 4.7). Yet, if the input data are not already well registered the merging could obviously fail. The method corrects the slight misalignments only in the normal direction. A tangential drift in the original registration could therefore cause a loss of sharpness or a loss of small details. Nevertheless, this degradation seems to pass unnoticed. Indeed, for last generation triangulation scanners like the one used for the experiments in this chapter, the registration error is very small. For a point cloud with side-length $99mm$ the observed average point offset after merging was $0.081mm$, with standard deviation 0.012. The tangential offset could not be measured. The explanation of the relative visual success of the method is that even a tiny normal offset causes a dramatic change in triangles orientation,

(a) Merged Result (b) Poisson Mesh (c) Bilateral filter

Figure 4.12: Comparisons of the merging (a) with a level set reconstruction method ([KBH06]) of the unmerged scans point set (b) and a filtering of the unmerged scans point set (c). The level set method obviously introduces a serious smoothing, yet does not eliminate the scanning boundary lines. The bilateral filter, applied until all aliasing artifacts have been eliminated, over-smoothes some parts of the shape.

and therefore completely jeopardizes the visual quality of the triangulation. An equally small tangential offset seems to be visually undetectable. Thus, the merging method corrects the normal error, and makes the tangential error unnoticeable.

The proposed merging can be seen as a local non rigid registration. Therefore it can be compared to the result given by state of the art non rigid registration methods [BR07]. To perform the comparison, the problem arose that the scans did not systematically contain strongly identified features. Most scans of the mask point set were simply rejected by the non rigid registration method described in [BR07]. In order to perform a serious comparison anyway, two sweeps of the fragment 31u of the Stanford FUR project were used. The computation times were, however, considerably different: it took more than $2h30$ to register non rigidly these meshes. On the same computer, using only the raw points and not the meshes, the merging took only $84s$. The final meshes were built using Poisson Reconstruction [KBH06] in both cases. The registration artifacts (two horizontal lines limiting the overlap area, fig 4.13) are much less visible with the scan merging than with the non rigid registration.

4.5 Conclusion

The main conclusion of the study is that it is possible to fuse multiple raw scans with minimal accuracy loss, provided an accurate previous registration has been performed. Future work will focus on the detection and handling of remaining holes, and on the automatic assessment of surface quality to replace a visual inspection.

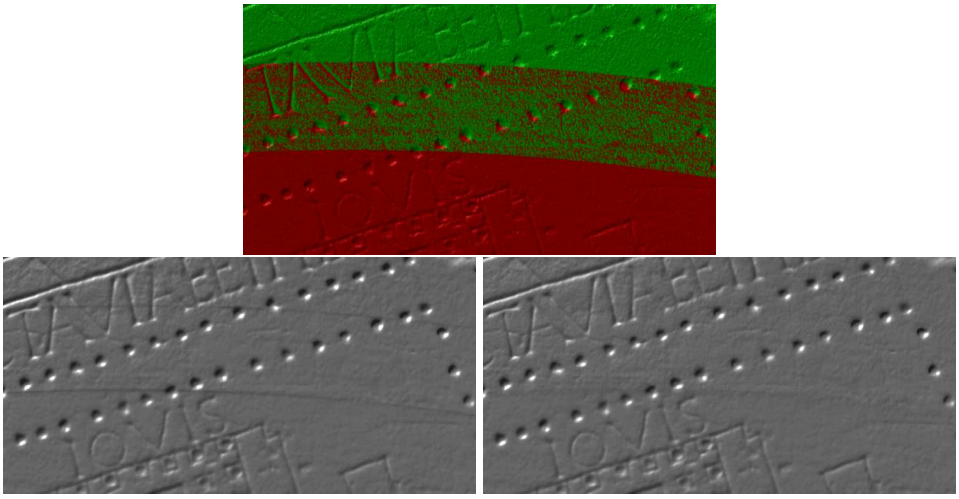
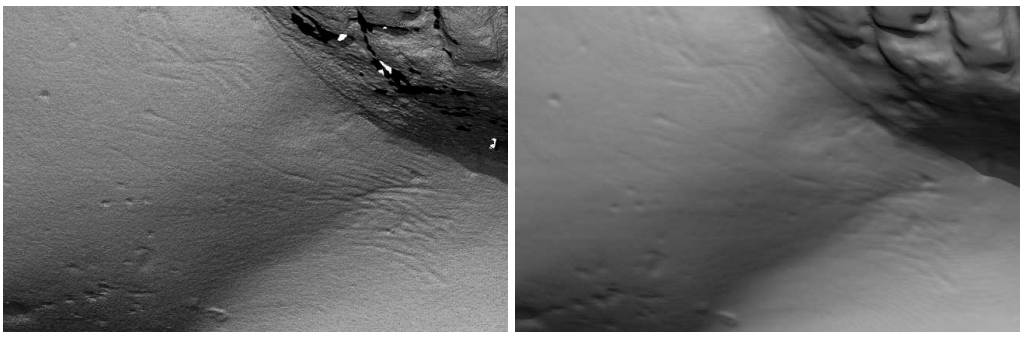


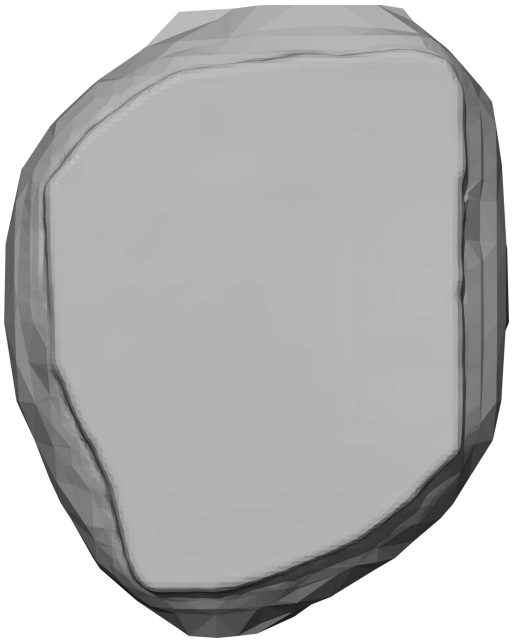
Figure 4.13: Comparison of registration of two scans (colored in different colors on the first figure) using Global Non Rigid Alignment [BR07] and scale space merging. Meshes were reconstructed using [KBH06].



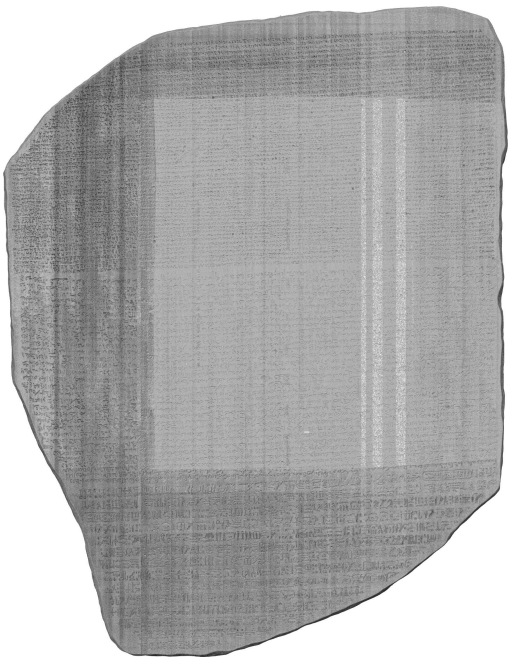
(a) Result of the merging (b) Result of the Poisson reconstruction

Figure 4.14: Comparison of the mesh obtained by merging and by Poisson reconstruction on a detail of nefertiti's cheek. In this case, Poisson Reconstruction suppresses registration artefacts but smoothes out the details.

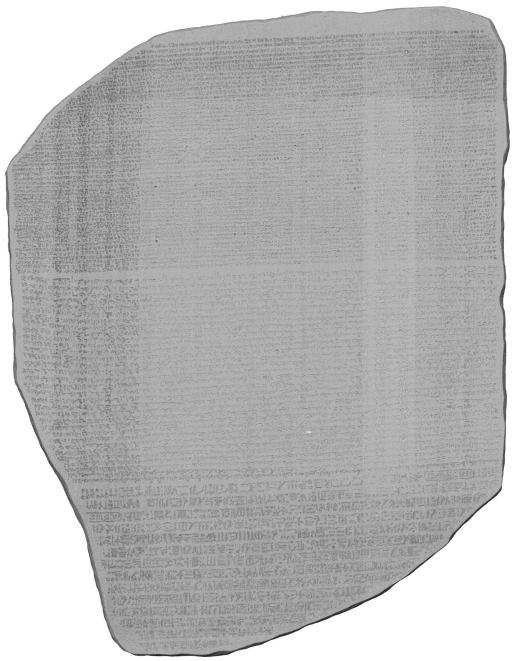
(a) Picture of the object



(b) Poisson Reconstruction of the input sets



(c) Without merging



(d) With merging

Figure 4.15: Comparison of rosetta meshes. A characteristic of this object is that the engravings in this object are very shallow (around 50μ), which is why Poisson fails. The artefacts in fig 4.15(c) are due to the 3D aliasing (this is fixed by scale space merging) but also to the resolution variation. Indeed the borders of the object were acquired using multiple orientations while the middle was acquired using only one orientation. This is why we have such a precision difference that is not fixed by the algorithm.

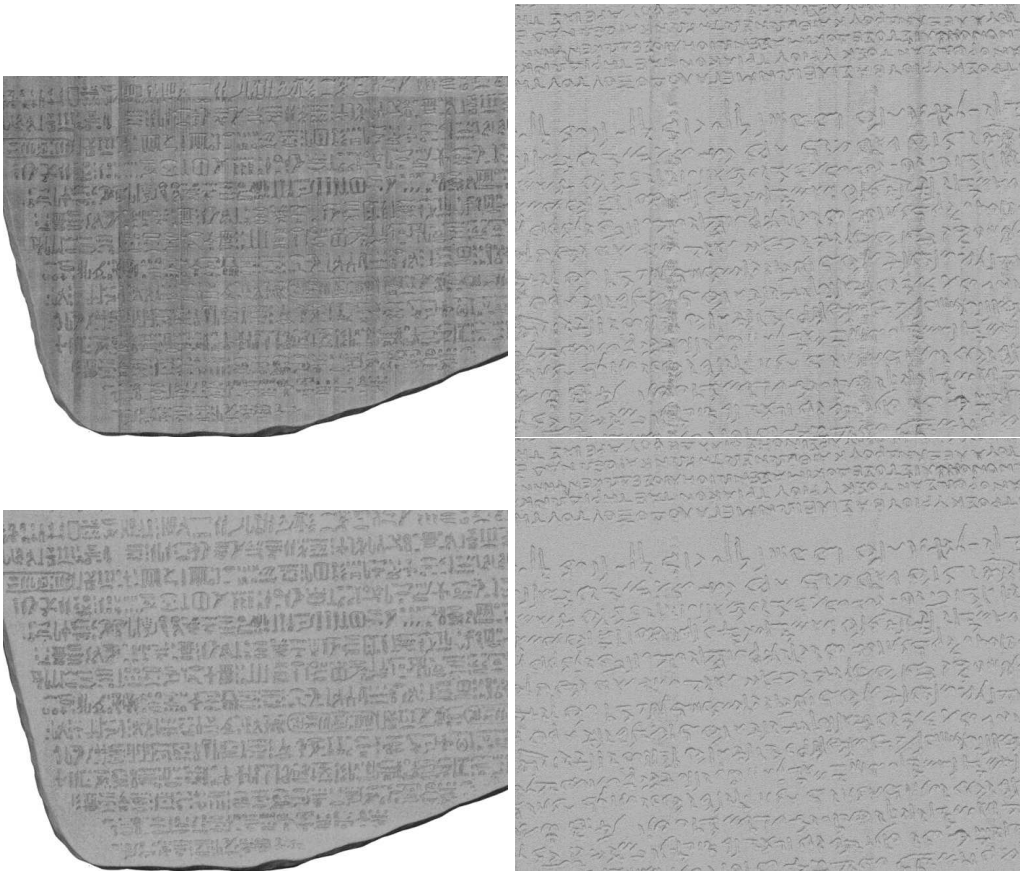


Figure 4.16: Details of the Rosetta object without merging (top row) and with fusion (bottom row)

Chapter 5

Filling Holes in Scale Space Meshes

Contents

Abstract: In this chapter, a method to fill holes in scale space meshes is presented. The method uses two key features of the scale space meshing method: first the fact that it permits to detect accurately the holes, second that it permits to work at a coarse smooth scale and then to transport back the result onto the meshed raw cloud. The goal is to fill holes so that the missing patches blend nicely with the rest of the object. The holes are first detected in coarse scale meshes, the filling patches are refined, and a mesh fairing method is applied so as to blend the patch in the surrounding mesh. The back propagating operator is eventually used to transport the patches to the fine scale mesh.

5.1 Previous Work

The problem of filling holes in meshes is crucial because all 3D acquisition systems have a limited access to hollow parts of objects. Even a very careful scanning process leaves behind a number of holes corresponding to shadowed parts of the object. Thus many solutions have been proposed to fill in the holes. First, one should note that meshes built by approximation of a signed distance function ([KBH06], [Kaz05], [ACSTD07] and [HDD*92] among others) automatically fill the holes, so that no postprocessing hole filling technique is required. This is actually a problem: The holes are not really detected. In consequence no exploration and no improvement of the quality of the restoration in these parts is possible. In agreement with the methodology developed in this thesis, given a mesh with holes the goal will be to fill in the holes without altering the other parts of the mesh.

The existing methods to actually fill in detected holes can be divided in two categories: the volumetric methods and the triangle patch methods.

5.1.1 Volumetric methods

These methods use ideas similar to the level set mesh reconstruction methods ([KBH06],[HDD*92]...) to determine in unknown areas (holes) a plausible surface.

In [DMGL01], the problem of filling holes is treated by building a signed distance function, whose zero level set is the surface. The goal is to preserve the geometry when it exists and to create smooth transitions to plausible geometry in unobserved areas. The surface is first converted to a volumetric representation: a 3D grid of values of a signed distance function defined only in a narrow band around the surface. An associated weight function measuring the confidence in the distance value is built. The distance values are diffused from known areas to adjacent unobserved areas. Once the diffusion is complete the zero set is extracted via marching

cubes ([LC87]). The diffusion is done by convolution with a low pass filter (see also [GLWYT06]). Many approaches based on a similar voxelization have been proposed. For instance [CBC*01] uses Radial Basis Functions (RBF) to approximate the distance and extract a watertight surface. The method in [VCBS03] extends the inpainting technique over the voxel grid to "inpaint" the implicit representation of the surface and extract the level set. [NT03] gives value 0 and 1 to voxels inside and outside the surface. Determining whether a cube is inside or outside the surface is done by projecting the shape onto k random planes. Each plane votes for the inside or outside position and a majority count yields the classification. In [Ju04] the cube labeling is deduced from the definition of a dual surface. In [PR05], besides the voxels labeling, cubes intersecting the surface are split in tetrahedra and the tetrahedra are also labeled. Then a graph-cuts algorithm is used to deduce the final labels.

In [ZGL07], a method that mixes patch filling and volumetric methods is presented. First holes are identified and filled with triangles using the advancing front mesh technique. Normals of the newly created triangles are estimated and used as vector field for solving a Poisson equation and obtaining the new coordinates of every vertex. Positions are then updated and the result is a smooth based patch.

An advantage of volumetric methods is that complicated hole topologies are not a problem. On the contrary patch filling methods mostly consider holes that do not contain islands. The next section reviews patch filling methods.

5.1.2 Patch filling methods

Detecting the holes in a mesh is a trivial task as their boundaries are closed loops of boundary edges (edges adjacent to only one triangle). All methods start by such a detection. Triangulating a 2D polygon is a problem with many solutions proposed in the past years. Nonetheless there is no simple method to fill a 3D polygon with triangles, indeed, [BDE96] proved that the problem of filling a 3D polygon is NP-complete. It also proved that if a perspective projection of a 3-dimensional polygon P from some point onto some plane Q is simple then P is triangulable. An algorithm for finding a suitable projection plane is proposed. For all pairs of edges, if the projection direction is inside the sphere region that sees the two edges intersecting, then the direction is invalid. A region of admissible projections can then be computed.

Many heuristics have been developed to fill the 3D mesh holes. [BS95] fills cracks in meshes by matching points on the crack boundary one with another and then stitching the borders with triangles (see also [TL94]). In [BNK02], vertex-vertex and vertex-edge correspondences are found on the hole boundary. The distance between the correspondences is used as an error to sort the pairs into a priority queue. The queue is processed in descending order: pairs are linked, new

correspondences created, and the priority queue is maintained up to date.

Since triangulating a 2D polygon is quite easy, the idea of projecting the polygon on a plane and triangulating has been extensively used. In [PSM*02], the surface is projected onto planes and standard image hole filling techniques are applied to the projection. Then the surface is back projected to 3D. Yet, a plane selection is possible because this method considers only range images. In [Jun05], a method was introduced to divide a complex hole (a hole that self intersects when being projected on a plane) into several simple sub-holes. If two projected edges intersect, the intersection points on both edges are found and projected back on real 33 edges. These points are used as splitting points and connected, separating the hole in two parts. Finally the inner mesh is refined to obtain regular triangles. In [LYZ08], a modification of [Jun05] is presented. The PCA regression plane is found and the hole is projected onto it. If the hole is not self-intersecting, it is simply filled. Otherwise, intersecting edges are found, and a parallelepiped-like region around the intersecting edge and orthogonal to the neighboring triangle is built. Inside this region, the algorithm looks for boundary points and selects one of them to create a new triangle adjacent to the orthogonal edge. The selection is done for example by comparing the angles between the initial triangle and the created triangle. The new triangle provides a natural splitting of the hole. In [BWS*09], a different method to unfold the hole boundary on a plane is introduced. The hole unfolding aims at making 4 consecutive vertices planar. Thus, for a loop of n points v_i, it minimizes the energy $E = \sum_{i=0}^{n-1} \angle t_i, t_{(i+1) \bmod n}$ where $t_i = (v_i - v_{i+1 \bmod n}) \times (v_{i+2 \bmod n} - v_{i+1 \bmod n})$. The minimization is done subject to the constraint that the minimum distance between two segments of the boundary loop should remain above a threshold ε. Simulated annealing is used to perform the minimization. After the unfolding the obtained polygon is triangulated by constrained Delaunay triangulation. The mesh is embedded in $\mathbb{R}^3$ by moving each of the boundary edge back to their original 3D position. The resolution of the inner mesh is then refined simply by standard Delaunay refinement algorithms. Inner vertices are added so that they minimize the area of the inner mesh. This is done by applying repeatedly a Laplacian smoothing.

Hole triangulation is not the only problem that arises. Indeed patches must blend nicely into the shape. For example in [PS96], holes are filled using splines.

In [Lie03] a popular geometric method for filling holes in meshes is proposed. Holes are filled by 3D polygon triangulation minimizing a given weight. Instead of using an area minimization weight, a weight that takes into account the dihedral angle between adjacent triangles is used. Thus, large dihedral angles are penalized. The patching mesh is refined so that its density matches the one of the surrounding mesh. Mesh fairing is applied to get a smooth enough surface.

In [PMV06], a topological grid is inserted into the hole (by mapping the grid of a disk onto the hole), and the inserted mesh is deformed to satisfy blending criteria. A bar network coupled to a geometric model is used to deform the inner mesh while

keeping the outside mesh identical. The mesh is deformed so that it minimizes the curvature variation while satisfying additional user-supplied constraints.

Only one of the patch filling methods explicitly addresses the problem of holes with islands: [LMW10] first determines the hole position and an "influence" area. If there is an island in the hole, the influence area and possible island is searched for feature points. Those feature points are linked accross the hole, creating feature curves that divide the hole into sub-holes. Each sub-hole is then filled. Yet the computation of the influence area is not explained.

Finally, Moving Least Square surfaces were also used to fill holes. In [WO07], planes are fitted to the set of the hole boundary points and a height for each boundary point is found. Finally new samples are interpolated by MLS interpolation on the height field. The use of MLS guarantees that new samples will fit smoothly into the mesh. Yet if the hole is not planar then it may fail dramatically.

Extending the idea of image inpainting has also been considered. The idea is that hole filling patches should look like other parts of the surface. The next section reviews the few surface inpainting methods that have been proposed.

5.1.3 Inpainting for surfaces, hole filling based on similarity

A review of mesh hole filling can be found in [BPK*08]. In [SACO04], the aim is to complete missing parts of the surface by transferring appropriately sampled and fitting regions of the shape. The surface is approximated by fitting a trivariate low degree polynomial. The space is divided into cells. A surface cell is called valid if it contains at least m points (there are enough points for surface representation in the cell) and invalid otherwise. For each invalid surface cell, the idea is to import and paste the content of a valid surface cell that matches the surface approximation in and around the invalid cell. The best matching cell is found by computing a vector valued signature for each cell and comparing them.

In [PMG*05], shape completion is made by combining geometric information from different context models. This is done by database retrieval, non rigid alignment of the candidate models and blending the aligned segmented models to get a consolidated model. For each missing region the best fitting model in the database has to be found. The shape is aligned to the shape database and information must be extrapolated from the models to region where data is missing. This is done by copying vertices from the model patches. The warping function is then recomputed for each model in the database to conform with the enhanced point set. After updating the alignment context patches are enlarged and the process is iterated until the hole is closed.

In [BSK05a] and [BSK05b], a multiscale inpainting method for surfaces is presented. Holes and a scale space is built from the input surface. By projecting the neighborhood onto its regression plane and resampling the neighbor's height on a

rectangular grid, a neighborhood descriptor is built. Thus the comparison between neighborhoods is easier. The 1-level inpainting is then defined as finding for every bounding point b an appropriate candidate point from the surface and to copy-paste its neighborhood. Multi level inpainting is done by considering the next level filling surface as a guidance surface for the filling of the current level surface.

[VCBS03] (see section 5.1.1) is also based on inpainting, yet the inpainting is performed on a implicit surface formulation.

5.1.4 Point clouds

Holes in point clouds have not been really investigated yet. A noticeable effort was made in [CJ03]. First, hole boundaries need to be found. In this work boundary points are characterized as being much farther from the barycenter of their neighborhood than other points. Since this would also detect points on sharp edges, another criterion is added taking into account the point distribution on the local tangent plane. Then a triangulation of the boundary curve is found and a set of points sampled on this triangulated surface is generated. In [BSK06], several criteria are combined to give to each point a boundary probability and extract boundary loops using this probability and coherence properties of the boundary.

5.2 A hole filling algorithm

5.2.1 Filling the hole: finding an initial patch

In [Lie03], an algorithm for finding the initial patch is proposed that aims at minimizing the angles between the patch triangles. Yet it requires storing a n^3 array for a hole with n border points. Some holes in our meshes contain more than 5000 border points, which makes this method intractable. Therefore, we shall use a simpler triangulation by projecting the contours on a 2D plane and detecting self intersections due to projections. This kind of triangulation can produce sharp angles between triangles. Better initial patch finding algorithms could be found, yet it is a first scale space mesh hole filling investigation. The proposed algorithm is summed up in Alg. 5. (The definitions of terms used in the algorithm are given below). Note that this algorithm works only for connected meshes. This means that the input mesh should contain a single connected component (the mesh is easily cleaned from remaining "islands" after singular triangles have been removed). This can lead to the loss of some samples.

A few explanations are necessary. Let a "border point" (resp. "border edge") be a point (resp. and edge) of a hole contour. Edges that are not border edges will be called inner edges.

Algorithm 5: Mesh Hole Filling

Data: A mesh with holes, each point is endowed with an oriented normal
Result: A hole-free mesh

```
1  Clean the mesh by removing all singularities (see definition below);
2  Find all boundaries as closed loops of boundary edges;
3  Remove the hole self intersections by splitting the boundaries ;
4  for each hole do
5      Add the hole to a list of subholes;
6      while subholes is not empty do
7          h ← first hole in subholes;
8          if the projected boundary does not self intersect then
9              Triangulate the 2D hole;
10         else
11             Split the hole in two subholes;
12             Add the resulting holes to the list of subholes;
13         remove h from the set of subholes;
14     Uplift the hole patch;
```

- **Singularities**: (line 1), all triangles having a inner edge with two border endpoints should be removed.

- **Connected mesh**: a mesh will be called connected if there is a path from one triangle to another composed of edge-adjacent triangles.

- **Hole contour**: Since we know the normal at each point position, we can easily find the triangle normal orientation. A contour point ordering is defined by setting that two successive points of the contour v_i, v_{i+1} adjacent to triangle $t = (v_i, v_{i+1}, v')$ with oriented normal $\ltimes(t)$ should verify: $(\overrightarrow{v'v''} \otimes \overrightarrow{v_{i+1}v_i}) \cdot \vec{n}(t) > 0$ where v'' is the projection of v' on (v_i, v_{i+1}).

- **3D self-intersection**: (line 3) In the case of manifold meshes, no edge can intersect with another edge except at the endpoints. In this case removing the singularities means checking that the hole border contains each point only one time. If this is not the case the hole is split into two holes at the intersection point.

- **Filling a 2D hole**: (line 9) There exist many methods triangulating a 2D polygon. Here we used Seidel's method ([Sei91]). It simply finds triangles that cover the interior of the polygon. Triangle vertices are vertices of the boundary polygon so that no inner vertex is added in the initial patch finding step.

- **2D self-intersection**: (line 8) When projecting a hole on its regression plane the boundary might self-intersect. Thus the hole should be split into parts that project without intersection. Call v_iv_{i+1} and v_jv_{j+1} the edges that intersect when projecting the hole boundary. Triangles (v_i, v_{i+1}, v_{j+1}) and (v_j, v_{j+1}, v_{i+1}). This two triangles split the hole boundary in two closed loops. At the end all the patches of the split holes, as well as the splitting triangles, are added back to the initial self intersecting 2D hole patch.

- **Uplift**: (line 14) Uplifting the patch only means creating a 3D patch from the set of 2D triangles. Each 2D point is the projection of a 3D point, thus we simply create a 3D triangle whenever the vertices projections are linked by a 2D triangle.

5.2.2 Refining the patch

The created patches have very large triangles (since no vertex is added) and are not smooth. Therefore in a second step the patch is refined. The same refinement method as in [Lie03] is applied. We briefly sum up this refinement in algorithm 6. The idea is to continue adding inner vertices, while the density of the patch is below the density of the surrounding mesh. To do that a density value σ is computed for each point (called the scale attribute), $\sigma(v)$ is simply the average length of all edges adjacent to v. Then for each triangle the centroid c is computed along with its estimated scale attribute $\sigma(c)$ as the average of the scale attributes of the triangle's vertices.

Our experiments led us to consider $\alpha = 1.75$ as a good value. As a matter of fact, we consider this value as fixed and will not change it anymore.

The algorithm makes use of a well-known algorithm which is the *edge-swap* or *edge-flip* method. Consider a patch composed of two triangles (pqr) and (pqs). Edge pq is adjacent to two triangles: (pqr) and (pqs). Flipping (swapping) pq means destroying those triangles and creating (prs) and (qrs).

Relaxing an edge (Algorithm 6, lines 10 and 13) means that for two adjacent triangles, we check that the non-mutual vertices are outside the opposite triangle circumsphere. If this is not the case and if an edge swap can fix this, the edge is swapped destroying two triangles and an edge and creating two new triangles and the swapped edge.

This edge relaxation is necessary to avoid triangles with sharp angles (though there will be triangles like this near the hole boundary, since we do not add any additional vertices).

Algorithm 6: Patch refinement

Data: A set of holes with patches
Result: Refined patches

```
changed ← true;
while changed do
    changed ← false;
    Compute the scale attributes of hole boundary points;
    for each triangle t = (v_i, v_j, v_k) in the patch do
        Compute the triangle centroid c and its scale attribute σ(c);
        if σ(c) < α‖v_m − c‖ and σ(v_m) < α‖v_m − c‖ for m = i, j, k then
            changed ← true;
            Split the triangles into (c, v_i, v_k), (c, v_i, v_j) and (c, v_j, v_k);
            Relax edges (v_i, v_k),(v_i, v_j),(v_j, v_k);
    if changed then
        while edges are swapped do
            Relax all interior edges;
            Update the list of triangles;
```

5.2.3 Giving shape to the patch

Finally the patch should blend nicely into the surrounding surface. Therefore a standard mesh fairing is applied to the patch inner vertices so that the final mesh is smoothed. [Lie03] proposed to minimize a thin plate spline energy to obtain $\mathcal{C}^2$ continuity. This approach was first proposed in [KCVS98]. The idea is to minimize the energy:

$$\mathcal{E}_{TP}(f) = \int f_{uu}^2 + 2f_{uv}^2 + f_{vv}^2.$$

Let the umbrella operator $\mathcal{U}$ be defined for each vertex v by

$$\mathcal{U}(v) = \frac{1}{\#\mathcal{N}(v)} \sum_{v_i \in \mathcal{N}(v)} (v_i - v)$$

where $\mathcal{N}(v)$ is the one ring neighborhood of v.

[KCVS98] showed that this umbrella operator is a discrete analogon to the surface Laplacian.

Recursively this gives

$$\mathcal{U}^2(v) = \frac{1}{\#\mathcal{N}(v)} \sum_{v_i \in \mathcal{N}(v)} (\mathcal{U}(v_i) - \mathcal{U}(v))$$

A way of solving for the thin plate energy minimization is to iterate the following

$$v' = v - \frac{1}{\nu}\mathcal{U}^2(v)$$

where $\nu = 1 + \frac{1}{\#\mathcal{N}(v)} \sum_{v_i \in \mathcal{N}(v)} \frac{1}{\#\mathcal{N}(v)}$.

This leads to an inflation or deflation of the patch to match the normal of the boundary triangles.

5.2.4 Special holes

In some cases we do not want to fill the holes, in particular the base holes (the base of a statue for example). This is why in some cases we exclude the base from the set of holes to fill. This base is detected by finding the hole with lowest coordinate in one specified direction. In the case of our acquisition device, this direction was always z.

5.3 Results on synthetic data

5.3.1 Sphere example

The following is an implementation of Liepa's method ([Lie03]) with the single difference that a different initial patch finding algorithm is used as discussed in section 5.2.1.

Figure 5.1: Initial hole (left), intial patch (middle) and refined patch (right).

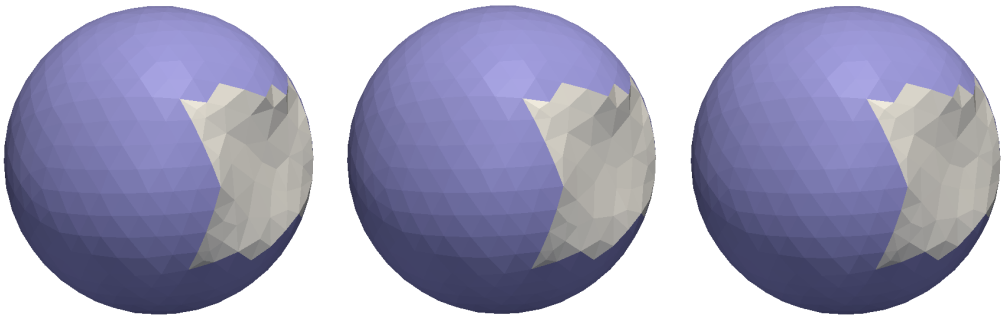


Figure 5.2: Thin plate spline minimization iteration 1,2 and 3.

Figure 5.3: Thin plate spline minimization iteration 5, 10 and 50.

Figure 5.4: Thin plate spline minimization iteration 100, 500 and 1000.

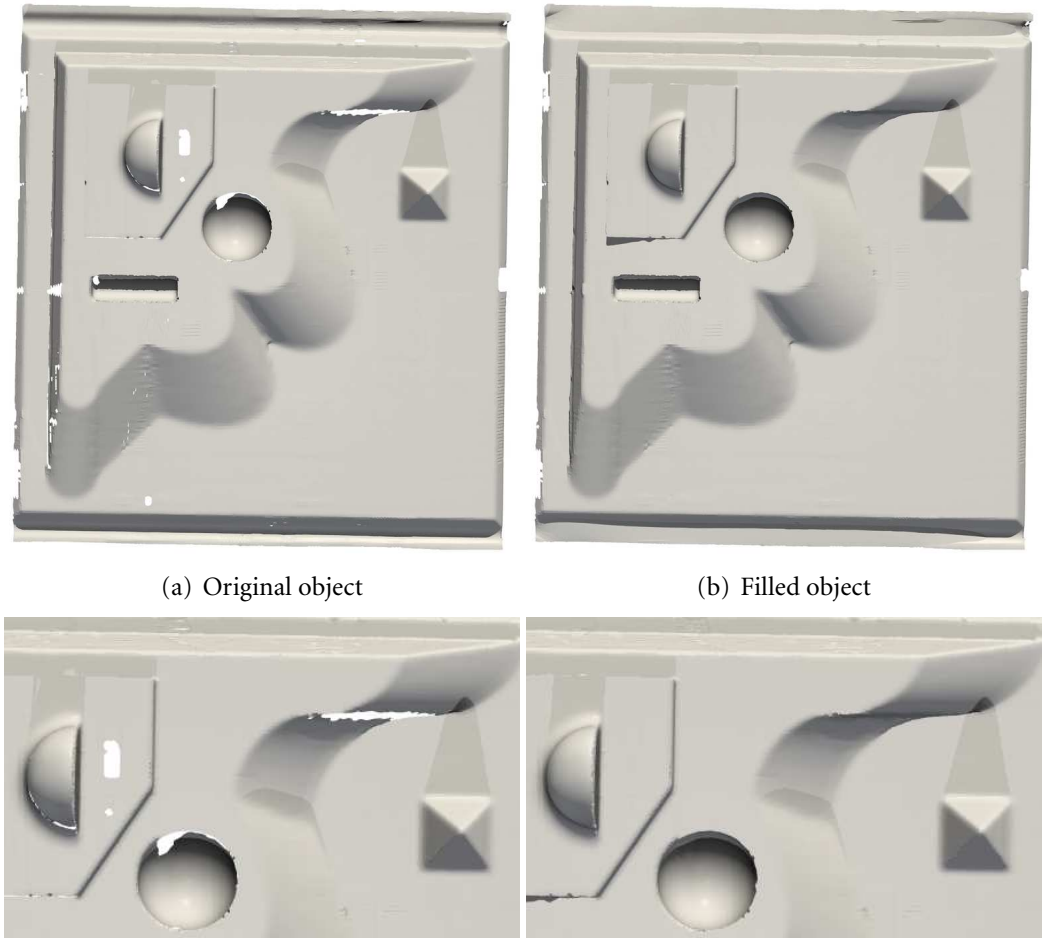
5.4 Scale space meshes

Scale space meshes are a special case. Indeed they have (at least) two resolutions: the coarse scale mesh and the fine scale mesh. At which scale should the hole filling take place?

Since the hole filling technique assumes a $\mathcal{C}^2$ continuity, it seems wiser to apply the hole filling to the coarse scale mesh and then to use the back propagating operator to go back to the fine scale. Yet no such operator is defined on the hole filling patches.

Each patch point is consistently oriented with the rest of the mesh (this normal computation and orientation is done when adding the vertices). An estimation of the fine scale variations is easily obtained by the scale space decomposition: the amplitude of the first iteration motion gives exactly the local texture and noise of the shape. If we assume that the missing part of the surface do not include any sharp feature then we can just add noise to the patch points in their normal direction. The noise addition is an estimation of the backward scale space operator for patch points. It implicitly assumes that holes do not contain sharp features but only fine scale variations.

The amplitude of the directional noise is gaussian and it has the variance mean of the first iteration motion. This is enough for the patch to "blend in" the rest of the surface. Figures 5.5, 5.6, 5.7, 5.8,5.9, 5.10,5.11, 5.12, 5.13 show results of the hole filling method. It appears that it works well for small holes, yet is not good enough to fill large holes satisfyingly (see the bottom of the mask, fig. 5.9).



(a) Original object (b) Filled object

(c) Detail of the original object (d) Detail of the filled object

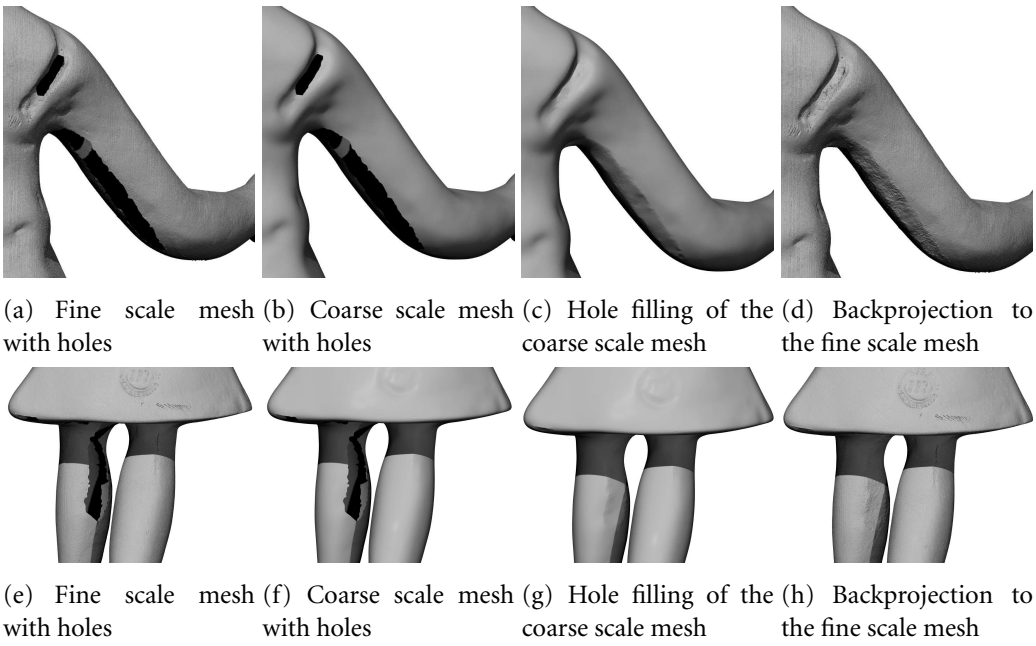
Figure 5.5: first test object, a mire (left) and the same object with filled holes (right), here the base is not removed

Table 5.14 gives the computation time for various point sets.

Conclusion

This chapter is clearly exploratory and it is not published. Its goal was to make a first review of hole filling algorithms and to define a first method to detect and fill holes taking advantage of the scale space meshing method. Several improvements are still necessary. In particular, a better initial patch should be found. An important improvement would also be to handle holes with islands.

Figure 5.6: Dancer with Crotales: fine scale mesh (left), fine scale mesh with filled holes (right) , here the base hole patch is removed



(a) Fine scale mesh with holes (b) Coarse scale mesh with holes (c) Hole filling of the coarse scale mesh (d) Backprojection to the fine scale mesh

(e) Fine scale mesh with holes (f) Coarse scale mesh with holes (g) Hole filling of the coarse scale mesh (h) Backprojection to the fine scale mesh

Figure 5.7: Dancer with Crotales: fine scale mesh (left), fine scale mesh with filled holes (right) . Here the base hole patch is removed.

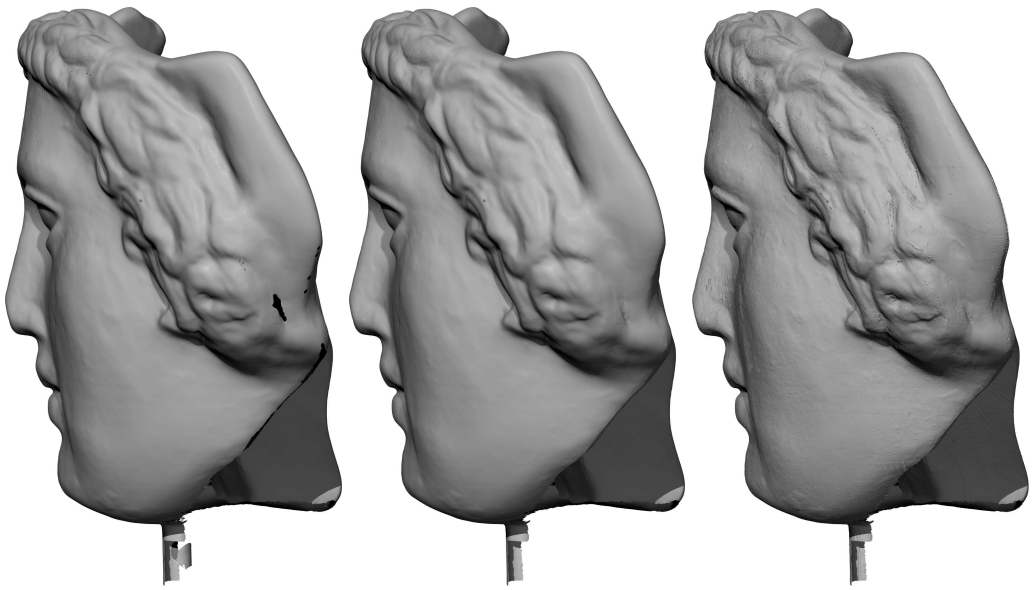
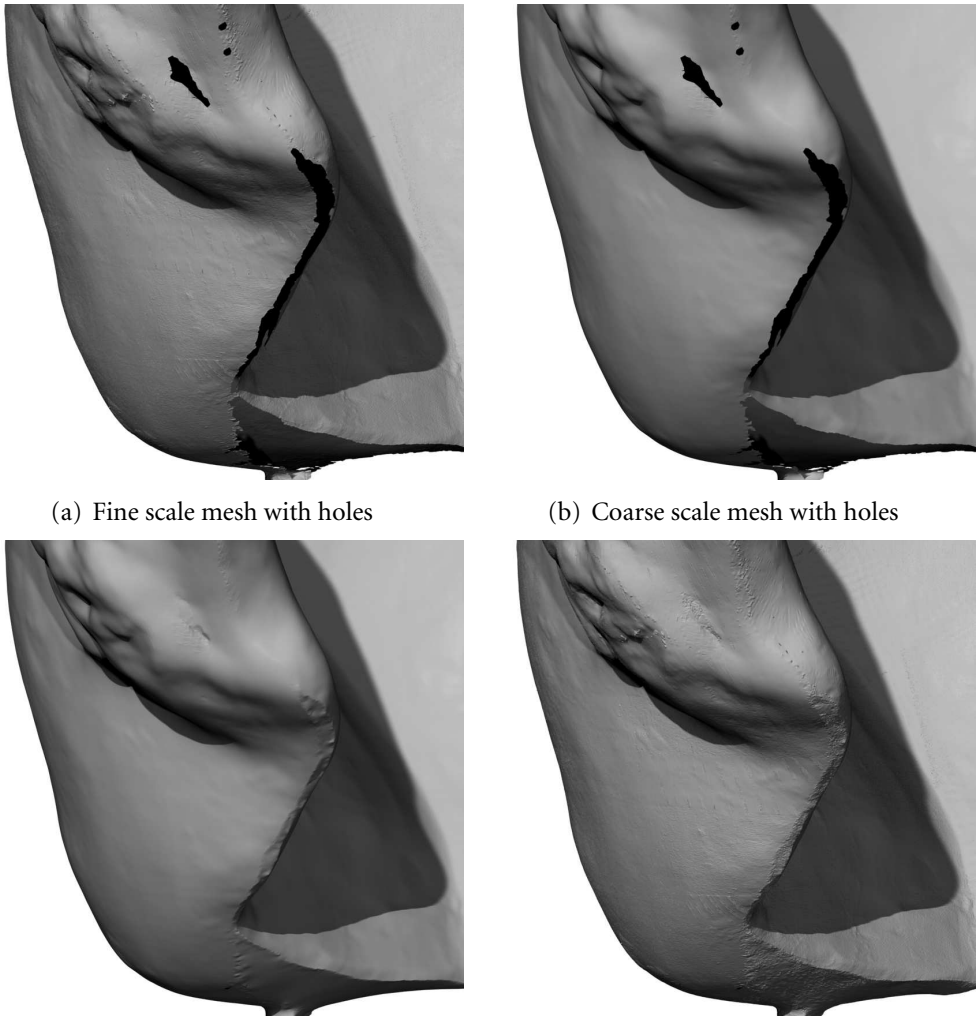


Figure 5.8: Mask: coarse scale mesh (left), coarse scale mesh with filled holes (middle), fine scale mesh with filled holes (right) . Here the base hole patch is removed.



(a) Fine scale mesh with holes

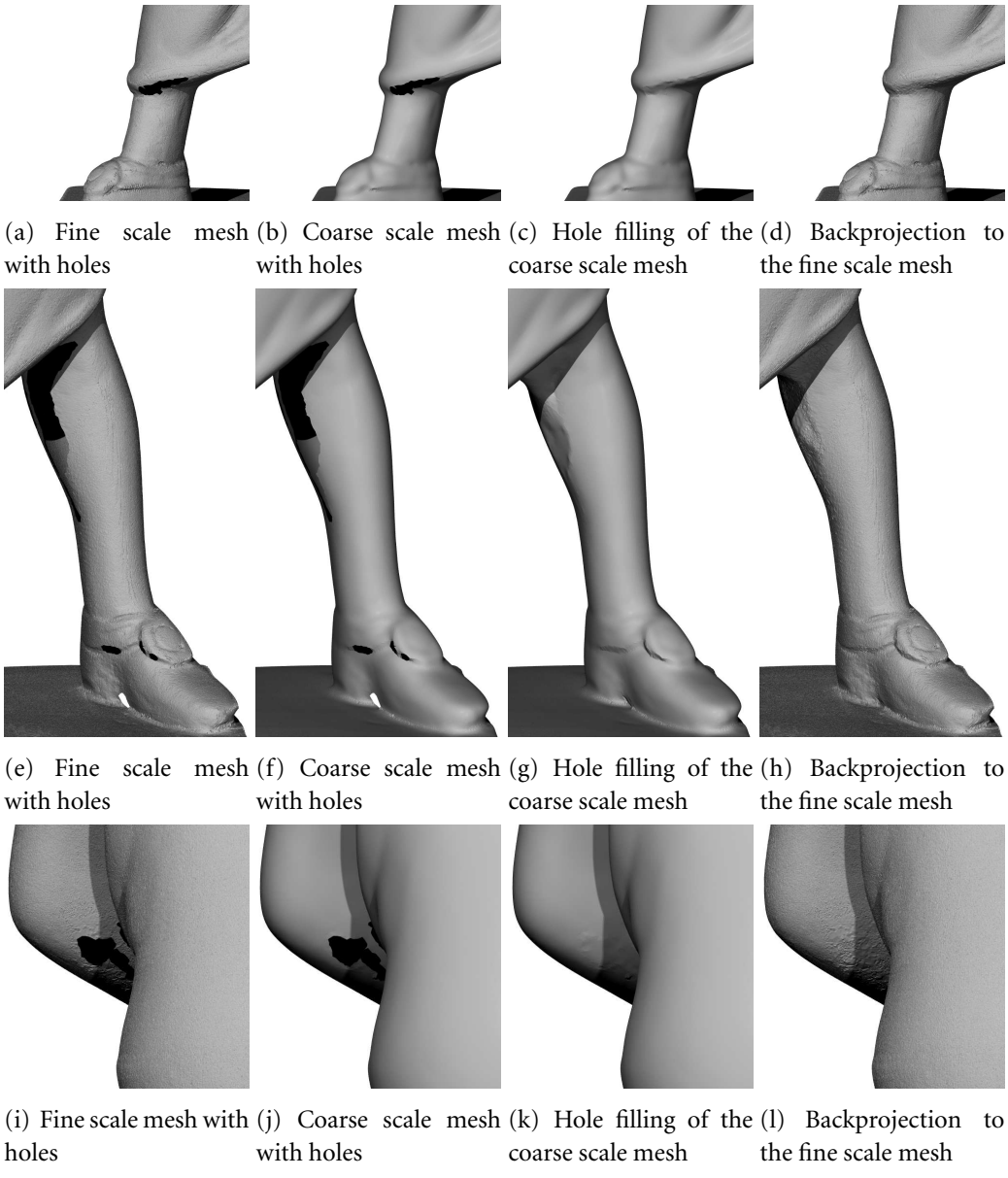
(b) Coarse scale mesh with holes

(c) Hole filling of the coarse scale mesh

(d) Backprojection to the fine scale mesh

Figure 5.9: Hole filling of the mask mesh (details)

Figure 5.10: Hole filling on the mime



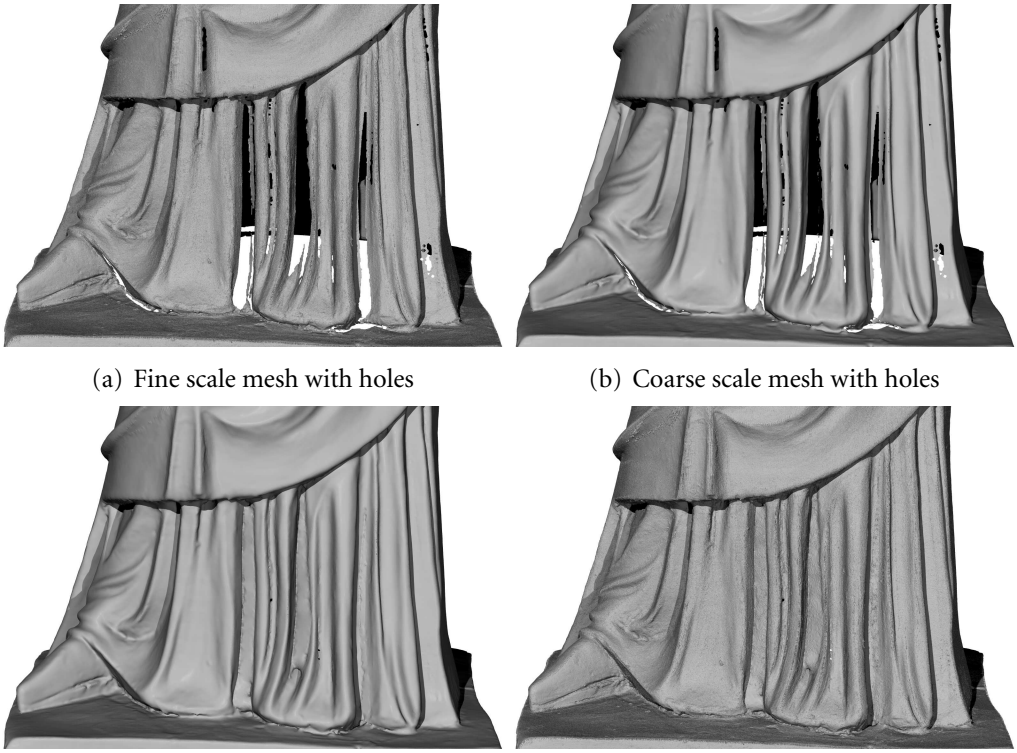
(a) Fine scale mesh with holes (b) Coarse scale mesh with holes (c) Hole filling of the coarse scale mesh (d) Backprojection to the fine scale mesh

(e) Fine scale mesh with holes (f) Coarse scale mesh with holes (g) Hole filling of the coarse scale mesh (h) Backprojection to the fine scale mesh

(i) Fine scale mesh with holes (j) Coarse scale mesh with holes (k) Hole filling of the coarse scale mesh (l) Backprojection to the fine scale mesh

Figure 5.11: Hole filling on the mime (details)

Figure 5.12: Hole filling on the Tanagra



(a) Fine scale mesh with holes (b) Coarse scale mesh with holes

(c) Hole filling of the coarse scale mesh (d) Backprojection to the fine scale mesh

Figure 5.13: Hole filling on the Tanagra (details)

Model	Number of points	Number of triangles	Number of holes	Computation time
Crotales	$5,496,386$	$10,988,071$	27	$246.13s$
Mask	$8,938,478$	$17,874,326$	108	$548.66s$
Mime	$8,571,061$	$17,139,634$	30	$270.08s$
Mire 1	$1,436,303$	$2,866,460$	117	$76.87s$
Tanagra	$16,211,676$	$32,385,734$	630	$37min20s$

Figure 5.14: Computation times for various models . All computation times are given for filling the holes in the fine scale mesh

Chapter 6

The Level Set Tree on Meshes

Contents

Abstract: Given a scalar function defined on a meshed surface, its level set component tree can be computed by a fast algorithm. This tree structure allows for an adaptation to meshes of the Maximally Stable Extremal Regions (MSER) method. Applied to the mesh curvature, this algorithm extracts significant curvature level lines and segments 3D surfaces into smooth parts separated by curves with high curvature. Segmentation results are shown on high resolution meshes of archaeological and industrial pieces. They compare favorably with MSER segmentations of pictures of the same objects.

6.1 Introduction

Segmenting meshes or point clouds into their significant parts is a basic but still challenging problem. Robust shape descriptors are required for most applications such as facet classification, shape registration, mesh simplification or shape retrieval. To this aim, the "crest lines" detection plays a role analogous to edge detection in image analysis. Crest lines are usually defined as the loci of directional extrema of curvature. Ridge (resp. valley) points can be defined as points where the maximum principal curvature takes a positive maximum (resp. negative minimum) along the line tangent to its eigenvector.

This notion being closely linked to the surface curvature, robust curvature estimators are needed. Curvatures can be computed by surface regression [CP03], by discrete schemes using the mesh triangle geometry [MDSB02], or by computing the surface tensor [Rus04]. One of the most used curvature estimation method is [Tau95a], where curvatures are estimated by drawing curves on the mesh surface (see [LP05] for an adaptation to point clouds). Other mesh based methods use the angles between adjacent mesh triangles to determine the curvature [DVVR06],[HMG00].

Most crest extraction methods detect and link the points where the derivative of the curvature crosses zero. These methods are usually defined on meshes, but can be adapted to raw point clouds [LFM96],[BA05],[YBS05],[OBS04], [DIOHS08],[HPW05]. In [DIHOS07], potential features are extracted by regressing surfaces near those points, estimating the number of fitted surfaces, and deducing the feature type. The idea of using feature lines for surface segmentation was investigated in [SF04] and suggested in [IFP95]. Regression free methods include [GWM01], where the analysis of the surface local covariance matrix leads to point classification, or [PKG03] where a multi-scale approach is introduced, yielding good results for mechanical shapes. Indeed, texture features are detected at fine scale, whereas at coarse scale the features describe shape geometry and are more robust. Recent research has also made huge progress in the rendering of viewpoint

dependent apparent edges [DFRS03], [JDA07].

Crest lines have, however, two limitations. The first is that these lines are often obtained by computing degree three derivatives, which is not an easy task for noisy or textured surfaces. The second is that crest points must be connected by some heuristic linkage. The final crest lines being often open or broken, they do not provide a surface segmentation.

Analogous problems arise in image analysis with *edge detection*. Image analysis therefore also uses *segmentation* methods dividing the image into regions separated by closed curves. One of the most reliable ways to define such closed curves is to extract the contrasted level lines. The level lines are the boundaries of connected components of upper (or lower) level sets. They inherit from these connected components an inclusion tree structure (fig. 6.1). This structure was called *Tree of Shapes* in [BCM01], and a fast algorithm computing them, the *Fast Level Set Transform* is given in [MG98].

For an image I defined on a domain $\mathcal{D}$ and with values in $\mathbb{R}$ the level sets with level λ are

$$F^\lambda = \{x \in \mathcal{D} | I(x) \geq \lambda\} \text{ (upper level set)}$$

$$F_\lambda = \{x \in \mathcal{D} | I(x) \leq \lambda\} \text{ (lower level set)}$$

If $\lambda' > \lambda$, then $F^{\lambda'} \subset F^\lambda$ and each connected component of $F^{\lambda'}$ is contained in one connected component of F^λ. The set of connected components of upper (resp. lower) level sets partially ordered by inclusion is therefore a tree. The shape tree proposed in [MG98] is a fusion of both trees.

The level sets can be represented by their borders ∂F^λ and ∂F_λ which are unions of closed Jordan curves, the image *level lines*. Several methods have been proposed to select the relevant level lines. A definition of meaningful level lines is given in [CMS05], [DMM01]. More recently the MSER method introduced the same objects with different names: the connected components of upper or lower level sets are called *extremal regions (ER)*. The ones with best contrasted level lines are called *maximally stable extremal regions (MSER)* [MCUP02]. The extraction of significant level lines to segment data is so useful that it has been extended to 3D medical imaging to extract meaningful level surfaces [CSA00],[MZFC09], and to video analysis [DB06], where extremal regions are tracked from frame to frame.

To the best of our knowledge these level line techniques have not yet been extended to meshes. The reason could be the lack of straightforward intrinsic scalar functions linked to a mesh, (such as the grey level for images). But there are actually such functions on meshes, the simplest one being the mean curvature. Several methods have already considered the curvature level lines and the umbilical points, but mostly from a theoretical point of view [KNSS09],[CPZ07] and [MWP96]. Until now, curvature level lines have not been studied as valuable feature lines, or used for surface segmentation. The goal of the present chapter is to describe an algo-

rithm computing all conspicuous curvature level lines, and to give experimental evidence that the method detects valuable mesh features.

In a way, the present work extends [KST09] and [ZTS09]. In these works, the surface is (implicitly) decomposed into a smooth base and a height value in the normal direction. Then the edges or iso-contours of this height are extracted. The Mesh-MSER framework considers this same situation in a more general setting : a surface with any scalar function defined on it. As shown on Fig. 6.9, results comparable to the results of [ZTS09] can be obtained by a significantly simpler and more general procedure.

The remainder of this chapter is divided as follows: Section 6.2 recalls the image MSER method, discusses its adaptation to meshes, and gives the algorithm building the level set component tree. Section 6.3 describes the algorithm extracting maximally stable extremal regions from this tree. Section 6.4 shows results on various simple and complex scanned objects, discusses strategies for level line selection, and compares 3D Mesh-MSER results to 2D MSER results on pictures of the scanned objects.

6.2 Mesh Extremal Regions

6.2.1 Definition of MSER for 2D images [MCUP02]

Let I be a real function defined on an image domain $\mathcal{D} \subset \mathbb{Z}^2$. MSER needs an adjacency relation A for elements of I, and usually chooses a 4 or 8 connectivity. The boundary ∂N of any set $N \subset \mathcal{D}$ is defined as $\{q \in \mathcal{D} \setminus N \mid \exists p \in N,\ pAq\}$. An *Extremal Region* N is a region such that for every $p \in N$ and every $q \in \partial N$ one has $I(p) > I(q)$ (maximum extremal region) or $I(p) < I(q)$ (minimum extremal region). To define *Maximally Stable Extremal Regions (MSER)*, consider a sequence $(N_i)_i$ of nested extremal regions ($N_{i+1} \subset N_i$). A region N_{i^*} in the sequence is maximally stable iff its *area change rate* $q(i) = \frac{|N_{i-\delta}| - |N_{i+\delta}|}{|N_i|}$ has a local minimum at i^*. The small variation $\delta > 0$ is a parameter of the method.

The detection of MSER proceeds by:

1. sorting pixels by intensity;
2. iteratively placing pixels in the image and updating the list of connected components and their areas;
3. selecting intensity levels that are local minima of the area change rate as thresholds, producing MSERs.

In a more formal way, the method uses the fact that upper level sets $F_k = \{p | I(p) \geq k\}$ are ordered by inclusion: $F_{k+1} \subset F_k$. One calls extremal region any connected

component of some level set. For each connected component N_k of the level set E_k, either $k = k_{min}$, in which case N_k is the whole image domain, or there exists a connected component N_i of the upper level set F_i such that $i < k$ and $N_k \subset N_i$. Thus the set of extremal regions is a rooted tree [BCM01] called (upper) *level set component tree*. Its dual tree is the (lower) level set component tree. The fastest component tree method seem to be [NC04]. It is its tree structure that allows the fast selection of MSERs [DB06].

Recently the question of the interest of MSER features was raised in [KZBB09]. An improvement of the stability was proposed that took the perimeter of MSER nodes into account. This improvement avoids favoring regions with rounder contours during MSER selection. A similar computation could be done here. Yet the simple extension proved to work well without this improvement.

We now adapt these definitions to the case of meshes.

6.2.2 Level set trees: an extension to 3D mesh surfaces

Let (V, T) be a set of vertices and triangles sampled from a 2-manifold $\mathcal{M}$ embedded in the 3D Euclidean space $\mathbb{R}^3$. Points $v \in V$ are linked to other points of V by edges forming triangles $t \in T$. We will assume that each edge is adjacent to either one or two triangles, so that each point belongs to at least one triangle. This means that there is no orphan edge and no orphan point, and that the mesh has no edge adjacent to three triangles. In other terms, the mesh is a manifold.

To define a level set tree we need a real function defined on the mesh. The function $H\colon V \to \mathbb{R}$ associating with each vertex v its mean curvature $H(v)$ is chosen as our example throughout the chapter. Mesh regions will be defined as unions of mesh triangles. A level set tree requires a topology and therefore an adjacency relation on the mesh. Two triangles will be called adjacent if they share an edge. With this definition, analogous to 4-connectivity on 2D images, two regions sharing a vertex but no edge are disconnected. The main differences with the two-dimensional case are that the mesh itself can be disconnected, and that it usually contains scanning holes. If the mesh is disconnected the component tree is a forest. The algorithm will process independently each tree. Section 6.3.3 explains how to handle scanning holes.

As for images, connected components of upper level sets can contain topological holes which are themselves connected components of lower level sets (Fig. 6.1). Monasse [MG98] therefore proposed to build a *shape tree*, which is a fusion of the upper level set component tree with the lower level set component tree. Since an upper component tree is faster to build, and since it is the appropriate object to perform MSER extraction on the mesh, we limit ourselves to the upper component tree.

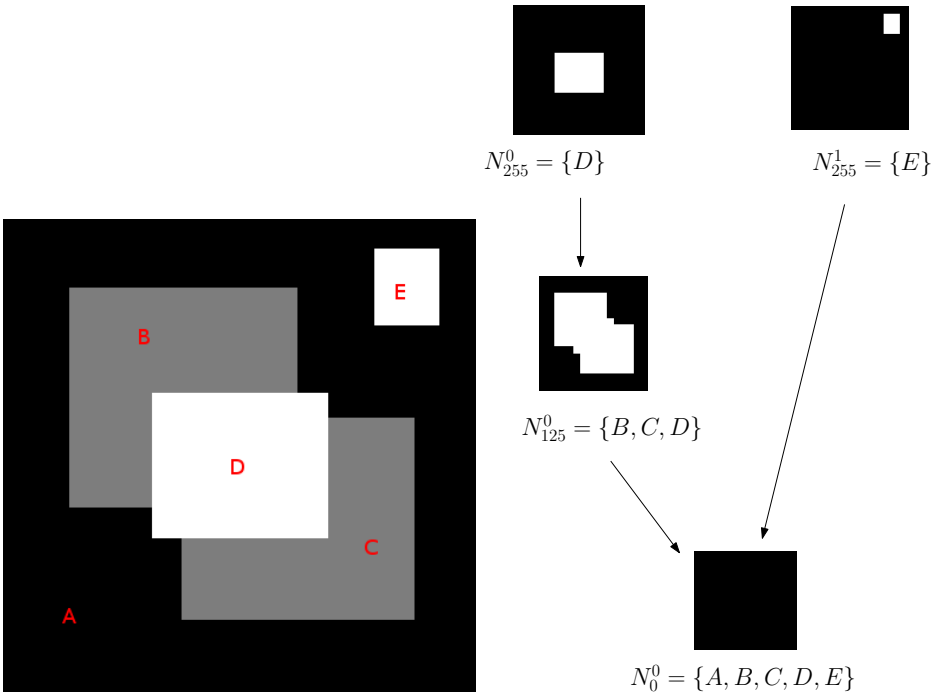


Figure 6.1: Example of an image (left) and its level set tree (right). Node A is the father of B and B contains a hole

6.2.3 Building the component tree for meshes

Set $\Lambda = \{h_1 < h_2 < \cdots < h_n\}$ a set of quantized values of H on V. The triangles $t \in T$ will be ordered by setting $Level(t) = \max\{k | \min_{v \in t}(H(v)) \geq h_k\}$. Referring to the 2D case, we can say that the triangles play the roles of pixels and that the real function used for building the component tree is $Level(t)$. Algorithm 7 describes the construction of the tree of level set components on T.

Algorithm 7: Building the Component Tree Forest

Data: A list of the mesh triangles F tagged with their levels
Result: The component tree forest $\{\mathcal{F}(\mathcal{M})\}$

```
Compute the levels of all triangles;
Sort F in decreasing level order;
Set all triangle markers to active;
for t ∈ F and t is active do
    k = t → level;
    Create an empty node N_new;
    E = {t};
    while E is not empty do
        Remove and return the first element t of E;
        Get t_1, t_2, t_3 the neighbors of t;
        for i = 1 ··· 3 do
            if t_i is active and t_i → level == k then
                Add t_i to the set E;
            if t_i → level > k then
                Get the node N containing t_i;
                Get N_a the last built ancestor of N;
                if N_a → level = k then
                    Merge node N_a into N_new;
                else
                    N_a.father = N_new;
                    Add N_a to N_new's children;
                |N_new| ← |N_new| + |N_a|;
        Mark t as inactive;
        Add t to the set of triangles of N_new;
        |N_new| ← |N_new| + Area(t);
```

Theorem 3. *Assume that the mesh (V, T) is a manifold (meaning that each edge is shared by at most two triangles). Then the list of nodes and father-child relations*

created by Algorithm 7 gives the graph of connected components of level sets of H a mapping defined on $\mathcal{M}$. This graph is a forest (meaning that the constructed graph has no cycles).

Sketch of Proof . For simplicity, in the algorithm and in the comments below, k denotes the level h_k. Only the order of the levels matters, and it is reflected by k. Algorithm 7 is strongly based on the triangle processing order from higher to lower levels. Indeed, while the algorithm is building nodes at level k, the only previously built nodes have a higher level. This fact is used for expanding the node: the expansion of a level k node is performed by successively adding the not previously added neighboring triangles of level k to the node triangle set. When expanding a new node N_{new}, we may encounter triangles already processed (line 14) and belonging to a node N at level l (then necessarily $l > k$; otherwise the node would not have yet been created). By going up in the parent-child relation, we can retrieve N_a (line 16), the last created ancestor of N (which can be N itself if it has no parent). For the same reason as before, N_a's level must be superior or equal to k. If N_a has the same level as k, then it belongs to the same node, and one can merge both nodes (line 18) by merging their triangle sets and adding N_a's children to the set of N_{new}'s children (and then deleting N_a). If N_a has a level larger than k, then N_a is a child of N_{new} (line 20), and we can set N_a's father to be N_{new} and add N_a to the set of N_{new}'s children.

During the construction of each node, track is kept of the triangles which are contained in this node, but not in its children. This way, each triangle belongs to a single node. The algorithm also keeps track of the area A (sum of the areas of triangles belonging to the node and to its descendants) while building the tree. This information is used in Mesh-MSER.

Algorithm 7 being similar to [NC04], the complexity of building the forest is also quasi-linear. Indeed, it starts by sorting the triangles, an $O(N \log N)$ step. When expanding a node, the computationally demanding case is when the encountered triangle belongs to an already created node. This requires finding the last created ancestor, which entails some traversing node operations, merge triangle lists (constant time) and add areas (1 operation). Since each triangle is processed only once, the total complexity is roughly $N \log N$.

There are as many trees in the forest as nodes with no parent. By arguments similar to [MG98] one can prove:

Theorem 4. *Any quantized scalar function H on a manifold mesh (V, T) can be reconstructed from its component tree. This means that no information is lost on H in the component tree.*

Assume the quantized levels of H are $h_k,\ k = 1, \cdots, n$. We want to extract local maximal elements (in the MSER sense) in the level set tree of H. Call $\delta > 0$

the level step used for computing the stability coefficient of a node N_{h_k} at level h_k. The set of test levels $h_1 < \cdots < h_n$ must therefore be complemented with all levels $(h_i + \delta)$ and $(h_i - \delta)$. For each node N_{h_i}, going up in the tree hierarchy yields its descendant $N_{h_i+\delta}$ with level $h_i + \delta$, and going down yields its ascendant $N_{h_i-\delta}$ with level $h_i - \delta$. Once these nodes are at hand, the stability coefficient is simply $q(N_{h_i}) = \frac{|N_{h_i-\delta}|-|N_{h_i+\delta}|}{|N_{h_i}|}$. For simplicity, we assumed in the previous relation that N_{h_i} only has one descendant with level $h_i + \delta$, which is not necessarily true. In the case of multiple descendants with level $h_i + \delta$, the sum of their areas is used instead of $|N_{h_i+\delta}|$. Remark that $N_{h_i-\delta}$ is not necessarily the father of N_{h_i}, since there is no condition on the size of δ relatively to $\min_i(h_{i+1} - h_i)$. For the same reason $N_{h_i+\delta}$ is not necessarily a son of N_{h_i}. This is why we must go up and down father-son relations in the component tree.

6.3 Extracting maximally stable regions from the component tree

6.3.1 Mesh-MSER: the algorithm

Algorithm 8 describes how to extract maximally stable extremal regions (MSERs). Starting from the component tree, this is an easy task. The tree structure gives a quick access to the area variations between $N_{h_i+\delta} \subset N_{h_i} \subset N_{h_i-\delta}$. When computing the stability coefficient, topological changes are authorized by the algorithm, whereas the original image-MSER technique only compares nodes on branches with no bifurcation. This way more lines are found than in the original 2D image method. The node merging procedure (13) is also standard. It generates a subtree whose nodes are exclusively the maximally stable regions. Merging a node $N_{h_{i+1}}$ into its father N_{h_i} requires a) removing $N_{h_{i+1}}$ from the set of N_{h_i}'s children, b) adding all of $N_{h_{i+1}}$'s children to N_{h_i}'s children, c) setting their father to be N_{h_i}, d) merging the list of triangles and e) updating the areas accordingly.

6.3.2 Triangle classification

Each selected node being given an index l, algorithm 8 yields a triangle classification L which, with any triangle t of the mesh, associates the label of the highest node containing the triangle (i.e., the label of the node with highest level containing the triangle). Because of the tree structure, it may occur that two triangles with label l are not connected by triangles with label l. Then the node must be split into different parts with different labels.

Algorithm 8: Mesh-MSER

Data: δ a step for computing stability coefficients and a set of levels $h_1 < \cdots < h_n$

Result: A labeling of the triangles and the borders of the detected MSERs

Build the component tree with levels $h_i \pm \delta$;
for *each tree node* N_{h_i} *(where* h_i *is the node level)* **do**
    Look for all descendants of N_{h_i} with level $h_i + \delta$ and the sum of their areas $A_{h_i+\delta}$;
    Look for the ascendant of N_{h_i} with level $h_i - \delta$ and its area $A_{h_i-\delta}$;
    $q(N_{h_i}) = \frac{A_{h_i-\delta} - A_{h_i+\delta}}{A(N_{h_i})}$;
for *each node* N_{h_i} **do**
    Get $q_{h_{i+1}}$ the minimal stability of the descendants of N_{h_i} with level h_{i+1};
    Get $q_{h_{i-1}}$ the stability of the N_{h_i}'s ascendant with level h_{i-1};
    if $q(N_{h_i}) < q_{h_{i+1}}$ *and* $q(N_{h_i}) < q_{h_{i-1}}$ **then**
        Select Node N_{h_i};
for *all non selected tree nodes* N **do**
    Merge the node N with its father;
Associate to each triangle the index of the node with highest level containing it;

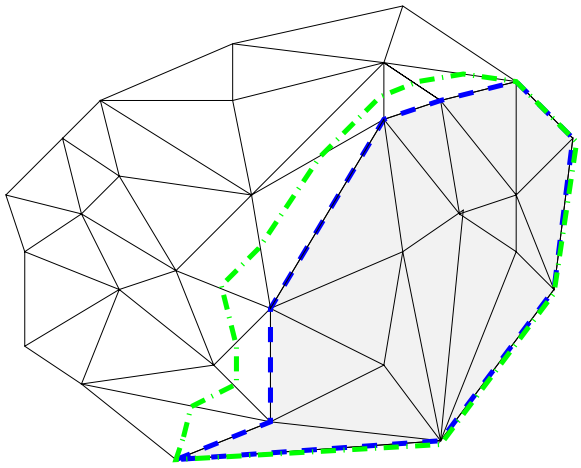


Figure 6.2: Extracting the border of a region (the region border with no interpolation at all is shown in the blue, and the interpolated region border is the dotted green line). Notice that a part of both borders coincide with the mesh border

6.3.3 Border extraction from selected region

Extracting the borders of each selected region is a simple task using the available list of triangles for each node. From this list one can extract those which share an edge with a triangle of lesser level. This yields a set of border edges which can be linked. Since we are dealing with meshes built on raw point sets with a triangulation method which does not fill in the scanning holes, the component border line may encounter a scanning hole. To extract the line, we first extract the set B of edges belonging to at least one triangle of the connected component which is either a hole border or a component border. Starting from an edge of B which is not a hole border, a line is extended by linking edges from B. This way only closed contour lines are built, which are not hole borders, but can partially coincide with hole borders.

6.4 Implementation and results

In the experiments herewith, the function H on the mesh will be the mean curvature. Choosing the levels h_i is another question. Since curvatures are real numbers and are estimated only up to a given estimation error depending on the curvature estimation method, using all levels for the h_i would not be a good solution. The adopted solution is to use equally spaced bins and δ equal to the quantization step.

The Mesh-MSER algorithm was implemented in C++. On a 1.5 GHz dual core laptop, without any particular effort on code optimization, the whole process lasts

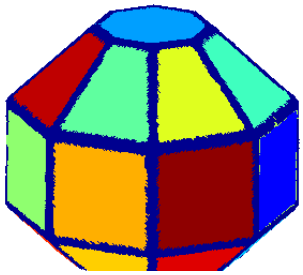


Figure 6.3: Classification by Mesh-MSER Analysis of a diamond shaped pattern (400k triangles).

for less than one minute for a 500000 vertices mesh. All shapes presented in this section (with the exception of the Stanford Urbis Romae pieces) were acquired using a triangulation laser scanner which yields a high precision dense point cloud (with some acquisition holes though, as can be noticed on fig. 6.4). The non-oriented point cloud was first oriented, its curvature computed at all raw points, and an interpolating mesh built using the method described in [DMMSL09].

6.4.1 Results on mechanical and geometrical shapes

The first experiment is a sanity check on simple a diamond shaped volumetric pattern (fig. 6.3) with $150k$ vertices and $400k$ triangles. Mesh-MSER surrounds all geometrically relevant areas, namely the facets and a single connected region containing all vertices. It could be objected that a single threshold on the curvature would have sufficed to obtain the facets. But, even in that simple case, it was not obvious to predict the right curvature threshold. Furthermore, a simple curvature threshold would have delivered many small extremal regions due to noise inside the facets, which are actually fused to the facets by Mesh-MSER. The second industrial example is a mesh acquired from a water pump (2.5 million vertices, 4 million triangles), whose the mesh was again built directly on the raw data. This object has many acquisition holes (see fig. 6.4). The final classification gives 200 regions. For better visualization random colors were given to the regions. The algorithm automatically separates edges from plane or curved parts. The segmentation of such a huge cloud into only 200 regions promises to enable a further model analysis, facet by facet.

6.4.2 Archeological pieces: comparative results

The next test (fig. 6.6) was performed on a good quality mould of an archaeological piece, which was subjected to a massive scan followed by Mesh-MSER. Mesh

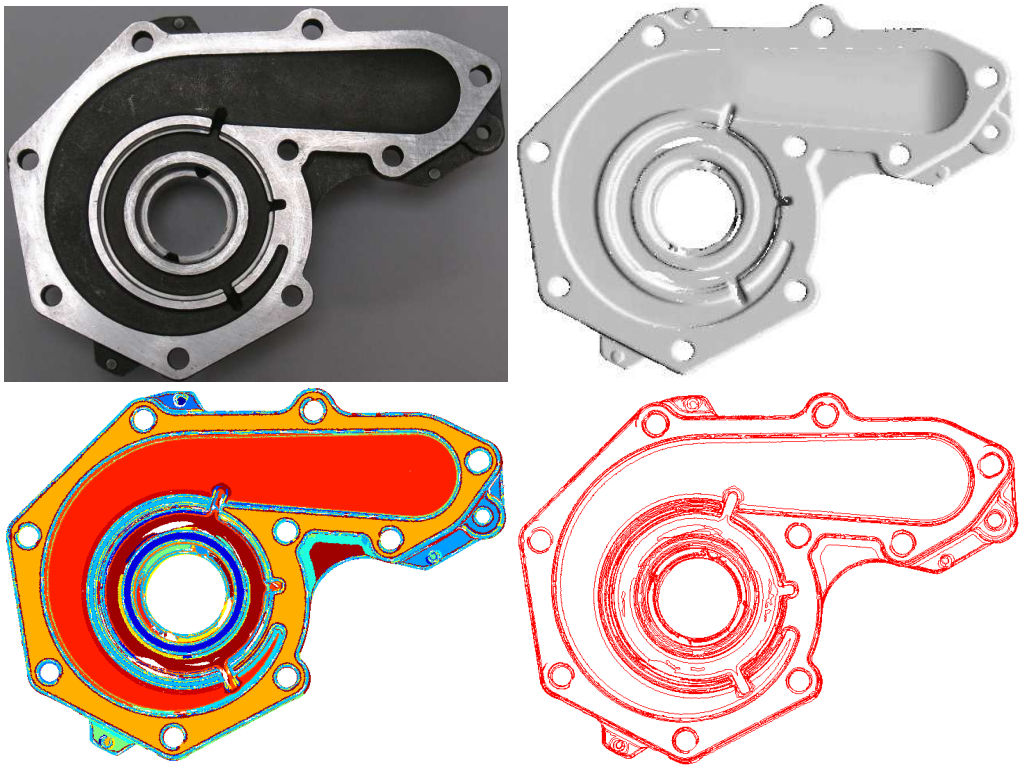


Figure 6.4: Classification by MSER Analysis (From top to bottom: picture of the object; obtained mesh; MSER segmentation; MSER borders)

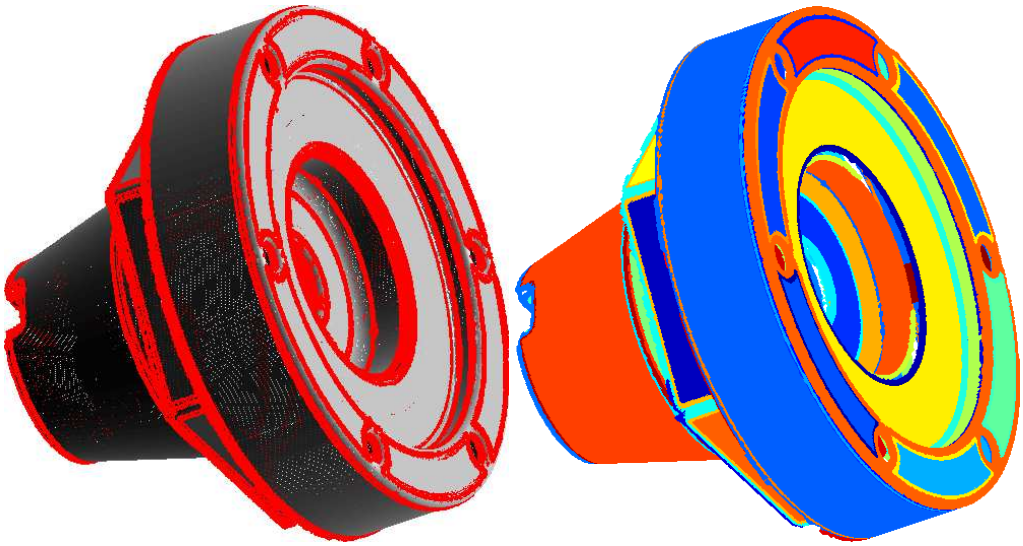
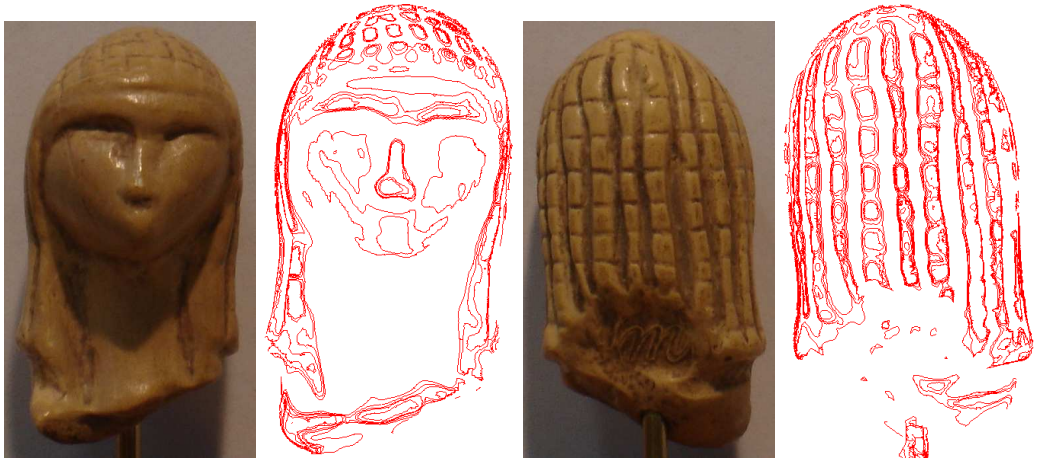


Figure 6.5: Line extraction (left) and segmentation (right) obtained by Mesh-MSER



(a) Picture (b) MSER (c) Picture (d) MSER

Figure 6.6: Maximally stable curvature level lines of "La dame de Brassempouy". Some lines on the above figure appear to be open because of the cropping into front and back part)

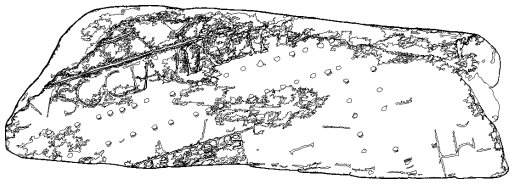
and curvatures were obtained using [DMMSL09]. This small prehistorical figurine (23.000 B.C.), "La Dame de Brassempouy" is only 2 centimeters tall. The mesh has approximately $300k$ vertices and $500k$ triangles. Notice how each detail of the hair dress is segmented out.

This preliminary exploration of the capabilities of Mesh-MSER was continued on the Stanford Forma Urbis Romae database, containing hundreds of archaeological artifacts coming from a broken stone map of Roma (see [KTN*06]). A challenge of this project is to solve the jigsaw puzzle and rebuild the map. It is a crucial test for the Mesh-MSER method to check whether or not it extracts the engraved symbols and drawings figuring the town map, and whether it does it better than what can be done with 2D-MSER or with a Canny edge detector from simple photographs. Pictures of fragments 10g and 31u are given along with the result of MSER extraction on figs 6.8 and 6.7. The experiment of fig. 6.7(b) shows Mesh-MSER working on these engravings with a high performance, comparable to the best 2D image MSER performance on pictures containing high contrasted trademarks and logos [MCUP02]. Indeed, all visible symbols and all features of the map plan are faithfully extracted, with very few outliers.

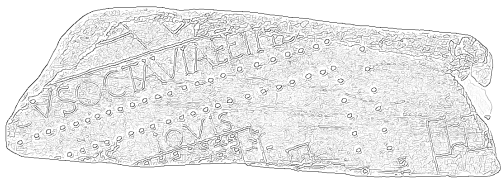
This experiment can be pushed further. Indeed, the pieces being rather flat, a direct comparison of 2D- and Mesh-MSER on their main facet makes sense. The Mesh-MSER result compares advantageously to 2D level line or edge extraction methods applied to a picture of the same object (fig. 6.7 (a)). The comparison shows that it is far more reliable to detect boundaries on the 3D mesh.

(a) Original picture of fragment 31u

(b) Obtained Mesh



(c) Meaningful level lines [DMM01] of the original picture



(d) Canny edge detector on the original picture

(e) Mesh-MSER result

Figure 6.7: Picture (a) of fragment 31u of Stanford FUR database, 3D mesh (b), Mesh-MSER result (d). This result can be compared with the 2D-MSER result (b) on a good quality picture of the same fragment [DMM01], and with the result of a Canny edge detector applied to the same picture (c). The comparison gives a sweeping advantage to Mesh-MSER. Indeed, 2D-MSER misses parts and keeps noisy level lines. Canny's detector has many outliers and yields anyway no segmentation. Similar experiments on artificial renderings of the mesh gave no better results

Finally, several strategies for extracting curvature level lines on meshes are compared on fig. 6.8 with fragment 10g of the Stanford Urbis Romae database. Here again the Mesh-MSER results seem to be complete, accurate, and without outliers. The experiment compares the choice of curvature level lines made by MSER with a simple threshold based on level line length. Although this choice indeed removes noisy level lines, it also loses many meaningful ones, and adds anyway a spurious threshold parameter.

To compare the obtained results with those presented in [ZTS09], the same data set point (a fragment of antique vase) was used. The mentioned reference is very similar in scopes to MSER: it proposes a two scale analysis on a mesh by defining a "base" and "height function". The lines shown in [ZTS09] are level lines of the height function. The base is implicitly defined by its gradient, by a sophisticated variational procedure. Here we used a similar height function to get a relevant comparison. The height function is defined as the difference between the surface and its smoothed out version by a large scale mean curvature motion. The Mesh MSER method can then be directly applied (fig. 6.9) on this scalar function.

6.5 Using the same approach with other functions

Other functions can be used for extracting valuable informations on surfaces. Let us for example consider the case of a digital elevation model. There are two ways of considering the data: either as an image and process it as gray-valued image using 2D-MSER or as a set of grid points with height values giving a set of 3D coordinates with a mesh structure. Then the process takes into account a much finer information since it adds precise triangle areas (which depend on the height) instead of adding a constant 1 area to compute node areas.

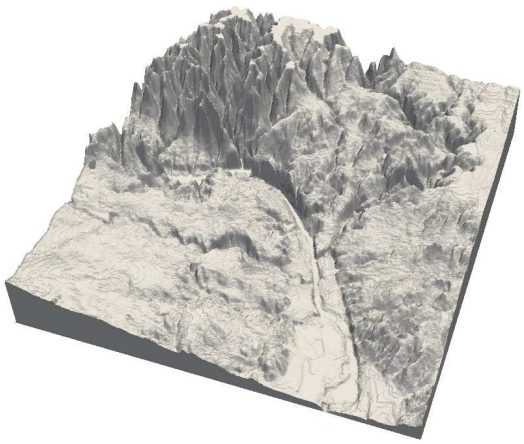
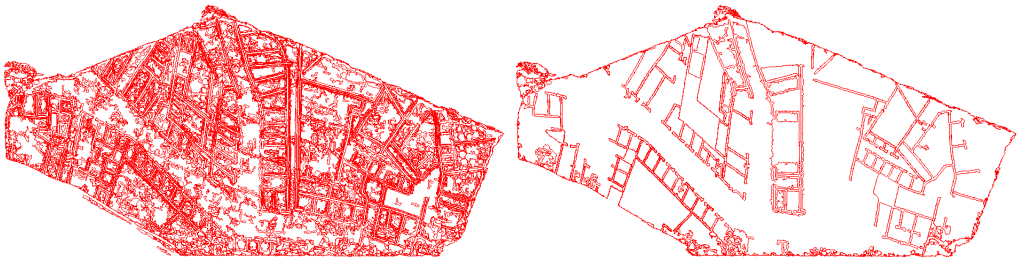


Figure 6.11: Rendering of a digital elevation model .

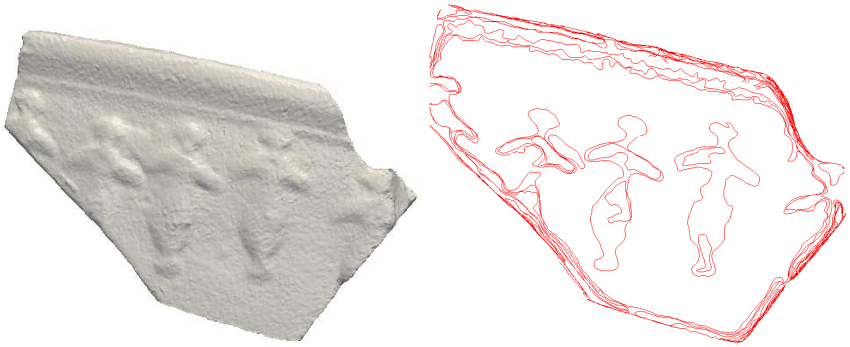
(a) Picture of fragment 10g (b) Obtained Mesh



(c) Selecting only lines with length above 100 (d) Selecting only lines with length above 10000

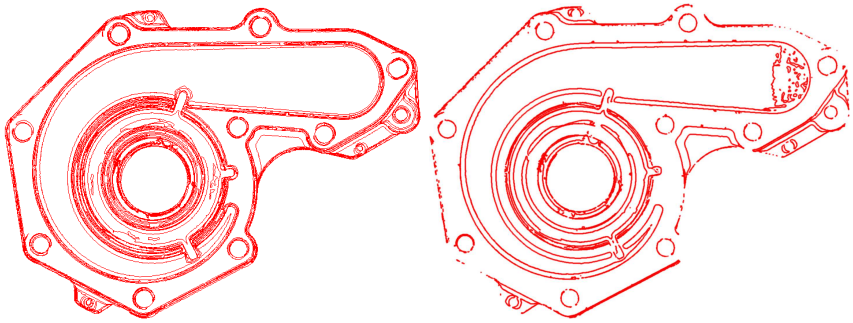
(e) MSER Selection

Figure 6.8: Comparison between several strategies for extracting level lines: a) Picture of fragment 10g of Stanford FUR database ; b) and c) level lines with length above a given threshold; d) Mesh-MSER: its selection is definitely much more accurate, misses no apparent detail and gives very few outliers



(a) Original object (b) Mesh-MSER selection

Figure 6.9: Result of Mesh-MSER on a vase model. Compare with results provided in [ZTS09] and [KST09]: results segment the shape into the relief and the base .



(a) Mesh-MSER Selection (b) Zatzarinni et al.

Figure 6.10: Result of Mesh-MSER on the pump mesh and comparison with the method by Zatzarinni et al. ([KST09]).

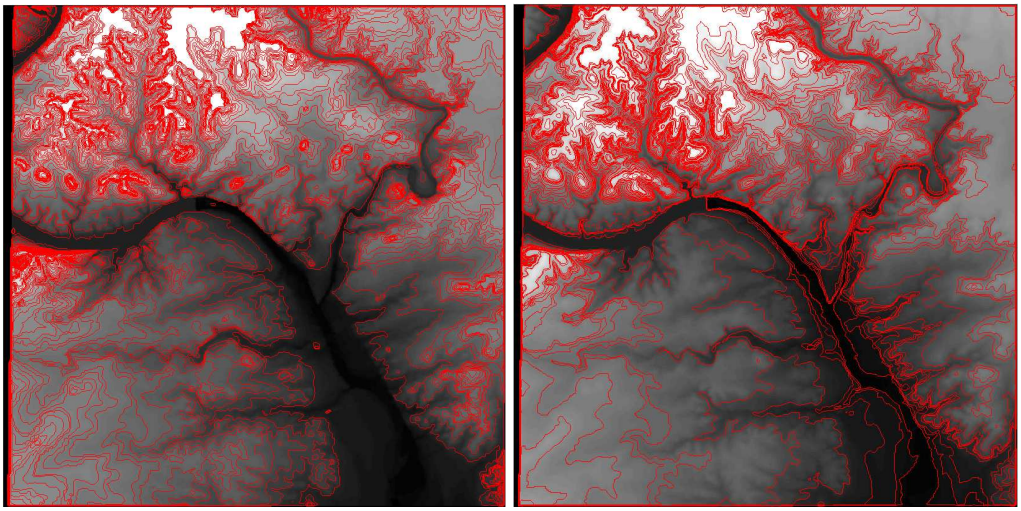


Figure 6.12: 2D-MSER on the gray-level image (left) and Mesh-MSER applied to the corresponding mesh (right)

6.6 Conclusion

This chapter introduced a fast level set tree method, Mesh-MSER, applicable to any scalar function defined on a mesh. This method is a direct extension of classic 2D image analysis tools building trees of level sets components or of level lines. Using the fact that the curvature is the most straightforward scalar function defined from and on a mesh, the method was used to segment meshes into maximally stable extremal regions (MSERs) of the curvature. Future work will focus on the exploitation of this structure. Indeed the experiments clearly point out the possibility of using the detected curves and regions to perform pattern recognition of complex objects such as the Urbis Romae fragments. On the other hand the method provides automatic segmentations of industrial objects into edge parts and parts with constant or slowly varying curvature, for which spline or conical models should easily be estimated.

Acknowledgements Pictures and scans of the Stanford Forma Urbis Romae are used with permission of Professor Marc Levoy (Stanford Digital Forma Urbis Romae Project). The vase model is property of Laboratory of Computer Graphics & Multimedia at the Technion and the Zinman Institute of Archaeology, University of Haifa.

Chapter 7

A numerical analysis of raw point cloud smoothing

Contents

Abstract: 3D acquisition devices acquire object surfaces with growing accuracy. Preserving this accuracy means conserving a very irregular sampling and requires also a numerical methodology to compute differential operators on irregularly sampled surfaces. Many such methods have been proposed for meshed 3D data. However, computing directly differential operators on the raw point clouds as they are acquired (e.g.) by triangulation scanners is crucial because this can be done before any mesh re-sampling and can in particular provide useful information for the meshing process itself. There are therefore several classic methods proposing to compute analogues of differential operators directly from the raw data point set. This chapter proposes a method to analyze and characterize these raw point cloud local operators. It reviews a half dozen basic discrete algorithms which have been proposed to compute discrete curvature-like shape indicators on raw point clouds. It shows that all of them can actually be analyzed mathematically in a unified framework by computing their asymptotic form when the size of the neighborhood tends to zero. Assuming that the underlying manifold is smooth enough, the asymptotic form of these operators is obtained in terms of the principal curvatures or of higher order intrinsic differential operators which characterize the discrete operators. This analysis, completed with numerical experiments, permits to classify the discrete raw point cloud operators, to rule out several of them, and to single out others.

Introduction

The output of laser scanners or any surface acquisition system is usually a set of points sampled with more or less precision on the surface. In some cases the result comes as a *mesh*, i.e., as a set of triangles linking points. In other cases, no such information is given and the machine data is simply an unorganized point cloud. In this chapter we focus on the mathematical processing of the surfaces defined as point clouds. We also explain in terms of differential operators what happens in some of the most common surface regularization processes.

The field of numerical surface analysis has been widely studied over the fifteen past years, due to the development of computer graphics. Yet, in most cases the starting surface representation is a mesh, for the simple reason that meshes are much easier to handle than a point cloud. Indeed, numerically processing a surface always involves finding the neighbors of a surface sample, and a mesh is oriented and has a surface topology. The neighbors of a vertex are all points linked by a specified number of edges to the center vertex. Yet in case of highly irregular meshes the

processing can be problematic because then, triangle areas vary a lot. The most common mesh reconstruction methods from a raw point cloud define a signed function over $\mathbb{R}^3$ representing the distance to the object, and then extract the 0 level set which approximates the object surface. See (e.g.) [CL96], [HDD*92], [CBC*01], [Kaz05], [ACSTD07], and the most popular such method [KBH06], which solves a Poisson equation to build the indicator field. These methods vary in the approach to compute the distance function, but all extract the 0-level set by using the marching cubes algorithm ([LC87], [KBSS01]). In such a meshing process, the initial raw points are irremediably lost, and this incurs into a loss of resolution. This explains why it may be relevant to process directly an unstructured raw point cloud. If no mesh is given as input data, then looking for neighbors means looking for points within a given distance of the query point. This neighborhood definition raises other problems: if the neighborhood is too large then we risk including two different parts of the surface in the neighborhood, and in the normal case of an irregular sampling empty neighborhoods can happen. Therefore the radius choice is crucial in the numerical implementation of any local operator.

The reminder of this chapter is divided as follows: section 7.1 reviews surface operators and motions previously defined on meshes and on point clouds, section 7.2 gives the necessary definitions and tools. Section 7.3 analyzes regression free curvature estimates. Section 7.4 and 7.5 analyze and compare the motions given by the projection on simple regression surfaces: a plane and a degree 2 polynomial surface. Finally, section 7.6 shows comparative numerical experiments.

7.1 Previous work

7.1.1 Curvature estimation and surface motions defined on meshes

Reviews for curvature estimation on surfaces can be found in [Rus04], [MSR07] or [MSR07]. Curvature tensor estimation methods were pioneered by Taubin [Tau95a] who presented a simple approximation for computing the directional curvature in any tangent direction. Then the curvature is computed in all incident edge directions and a covariance matrix of the edge direction weighted by their directional curvatures and the area of the two incident triangles is built. Eigenvectors and eigenvalues of this covariance matrix yield a simple expression of the principal curvatures and curvature directions.

Other curvature tensor estimation methods include [Rus04] where the tensor is estimated by building a linear system over the tensor coefficients. This linear system expresses the constraints that multiplying the tensor by an edge direction should give the difference of the edge's endpoints normals. The same method is

applied to find the curvature derivatives. Normals are also used in [TRZS04] to give a piecewise linear curvature estimation (see also [TT05])

To avoid computations of derivatives, a new kind of curvature estimation method has been proposed, based on domain integration. In [YLHP06], [PWY*07] and [PWHY09], the intersection of the surface with either a sphere or a ball centered at a vertex is analyzed: the covariance matrix of this domain is computed and eigenvalues are expressed in terms of principal curvatures. By increasing the neighborhood radius, the curvature estimate can be made multiscale. A very interesting feature of these methods is that they do not rely on high order derivatives and are therefore more stable. Other methods to compute the curvature include the normal cycle theory [CSM03].

Surface motions have also been studied as a part of a mesh fairing process. A key method was introduced by Taubin in [Tau95b] who considered a discrete Laplacian for a mesh V with vertices v_i $\frac{\partial V}{\partial t} = \lambda \mathcal{L}(V)$, $\mathcal{L}$ being a discretization of the Laplacian $\mathcal{L}(v_i) = \frac{1}{\text{card } N(v_i)} \sum j \in N(v_i)(v_j - v_i)$ where $N(v_i)$ is the set of vertices linked by an edge to v_i (1-ring neighborhood). This formulation is widely used. Indeed [DMSB99] uses a similar "umbrella" operator. [GH00] also computes the discrete Laplacian for all mesh vertices and the eigenvector and eigenvalues of the Laplacian are computed. By removing the smallest eigenvalues, a fair mesh is obtained.

A well known formulation of the Laplace Beltrami operator is the famous cotangent formula ([MDSB02]). This formula writes:

$$\Delta v_i = \frac{1}{2} \sum_{j \in \mathcal{N} v_i} (cotan\alpha_{ij} + cotan\beta_{ij})$$

where v_i is a vertex of the mesh, $\mathcal{N}_1(v_i)$ its one ring neighborhood, α_{ij} and β_{ij} are the angle opposite to edge $v_i v_j$ in the two triangles adjacent to $v_i v_j$. This has been used to compute the surface intrinsic equation.

7.1.2 Curvature estimation and surface motions defined on point clouds

We now discuss the rare approaches dealing directly with point clouds.

In [UH08], a scale space decomposition method for point clouds is introduced. At a point p, the intersection of the surface with a plane containing the normal to p is a curve. The curvature of these curves can be easily computed. An operator is defined that moves each point in the normal direction by a factor equal to this curve curvature. Averaging over all possible planes yields a mean curvature motion. The non-uniform sampling problem is then solved by using a density normalized kernel which removes the dependence of the result on variations in sampling density. The scale space is then used to select "scale-space extrema". At each scale the point

motion is considered. Introducing a scalar function on the displacement norm, the authors claim that they recover the characteristic scales of the surface (the introduced function is extreme for the characteristic scales).

More theoretical work which is not tested on real raw surfaces can be found in [MOG09] where a method to compute curvatures and normals based on voronoi covariance matrix is introduced.

In [Tan05], the proposed framework for curvature estimation at a particular point is based on a set of curves representing the local neighborhood of the point under consideration. By considering the neighbors of p, one can have the set of triplets (p_i, p_j, p) and those triplets can be used to define parametric space curves by quadratic polynomial interpolation with $p(0) = p_i$, $p(1) = p_j$ and $p(t) = p$ where $t = \frac{|p-p_i|}{|p-p_i|+|p_j-p|}$. This allows to approximate maximum and minimum curvature values as the minimum and maximum normal curvature values for all possible point triplets. This method can be used either on meshes or point clouds.

In [KSNS07], the authors propose a statistical estimation of curvature of point sampled surfaces based on IRLS (iteratively reweighted least squares) and M-estimators. Position difference vector Δp and normal difference vector $\Delta \mathbf{n}$ are used to define a linear system yielding a first estimate of the curvature tensor. Then residuals are computed and used to weight the samples and the objective function is minimized by iterative reweighting of point samples. This yields the final curvature tensor estimate. This procedure is rather time-consuming.

Finally in [BSW09], an algorithm to compute the Laplacian of a function defined on point clouds in $\mathbb{R}^d$ is proposed along with convergence proofs. Yet the model is not tested on real surfaces.

Neighborhood covariances being used already for normal estimation, the idea to express fundamental forms as covariances matrices was introduced. Next section reviews the different covariance techniques considered in the literature.

7.1.3 Curvature estimation using covariance techniques

There are few covariance approaches and none of them has been analyzed mathematically yet. Nevertheless, covariance methods can be an elegant alternative to surface regression. This chapter performs this analysis linking the various covariances to surface curvatures. Three papers use covariance matrices for curvature estimation: [BC94],[LT90] and [PGK02].

[BC94] considers the neighbors (p_i) of a point p. The second fundamental form is then defined as the covariance matrix of vectors pp_i projected onto the tangent plane of the surface at p. An equivalent of the Gauss map is also introduced: it is the covariance matrix of the neighbors unit normals projected onto the surface tangent plane at point p. The eigenvectors are said to give the principle directions. In fact these two covariance matrices are inspired from [LT90]. Indeed, [LT90] first

proposed to compute the covariance matrix of the normals of point p's neighbors, to extract the principal eigenvalues which correspond to the principal curvatures of the surface at p. The last covariance method, introduced in [PGK02], is not claimed to be explicitly linked to surface curvatures or fundamental forms, yet it is used to account for the surface geometric variations. Consider the covariance matrix of vectors p_i where is the barycenter of the neighborhood of p. The *surface variation* is then defined as the ratio of the least eigenvalue over the sum of all eigenvalues of this covariance matrix. This quantity has the nice property that it is bounded between 0 (flat case) and $1/3$ (isotropic case).

All these methods will be described in more details and analyzed in section 7.3. Another interesting technique for surfaces defined by point clouds is the Moving Least Squares Surfaces.

7.1.4 Moving Least Squares Surfaces

[LS81] first introduced MLS surfaces as follows. Given a data set of points $\{p_i\}_i$ (possibly acquired by a 3D scanning device), a smooth surface $\mathcal{M}$ based on the input points is defined. The goal is to replace the points p defining $\mathcal{M}$ with a reduced set $R = \{r_i\}$ defining an MLS $\mathcal{M}'$ surface which approximates $\mathcal{M}$. The surface given by the point cloud is expected to be a 2-manifold, C^∞ smooth. The authors also define a bounding error ε such that $d(\mathcal{M}, \mathcal{M}') < \varepsilon$, where d is the Hausdorff distance.

The projection of a point on MLS surface is defined as follows: given a point p, find a local reference domain (plane) for p. The local plane H is computed so as to minimize a local weighted sum of square distances of the points p_i to the plane. (Thus is is a weighted regression plane). The weights attached to p_i are defined as functions of the distance from p_i to the projection of p on plane H, rather than the distance from p_i to p.

Assume q is the projection of p onto H, then H is found by locally minimizing

$$\sum_{i=1}^{N}(< \mathbf{n}, p_i > -D)^2\theta(\|p_i - q\|),$$

where θ is a smooth, monotone decreasing function, which is positive on the whole space. We can set $q = p + t\mathbf{n}$ for some $t \in \mathbb{R}$, which gives

$$\sum_{i=1}^{N}(< \mathbf{n}, p_i - p - t\mathbf{n} >)^2\theta(\|p_i - p - t\mathbf{n}\|)$$

The local reference domain is then given by an orthonormal coordinate system on H so that q is the origin of this system. The reference domain for p is used to

compute a local bivariate polynomial approximation to the surface in a neighborhood of p. Let q_i be the projection of p_i onto H, and $f_i =< \mathbf{n}, p_i - q_i >$. In this local coordinate system, let (x_i, y_i) be the coordinates of q_i on H. The coefficients of the polynomial are computed by minimizing the least square error: $\sum_{i=1}^{N}(g(x_i, y_i) - f_i)^2\theta(\|p_i - q\|)$

The projection of p onto $\mathcal{M}$ is defined by the polynomial value at the origin, i.e., $q + g(0,0)\mathbf{n} = p + (t + g(0,0))\mathbf{n}$. Thus, given a point p and its neighborhood, its projection onto the MLS surface can be computed.

The approximation power of MLS surfaces was shown in [Lev98] and the first applications were introduced in [ABCO*01], [AK04] and [Lev03].

MLS surfaces turned out to be not only theoretically powerful but provided fine implementations for either rendering or up-sampling and down-sampling point sets ([ABCO*03],[PGK02]). Variants of MLS were proposed mostly for a better preservation of sharp edges in surfaces defined by point clouds ([OGG09], [LCOL07], [FCOS05]).

The same framework to build a scale space for point clouds in [PKG06]. The surface is evolved through a diffusion process $\frac{\partial p}{\partial t} - \lambda \cdot \Delta p = 0$, where p is a point of the surface, λ a diffusion parameter and $\Delta p = H\mathbf{n}$ is the Laplace Beltrami Operator (H is the curvature and $\mathbf{n}$ the normal at point p, this is the decomposition process). By remembering the set of displacements $D_i(p)$ of each point p we have a reconstruction operator. The choice of the Laplacian discretization is very important: a first possibility is to use the standard mesh Laplacian techniques ([Tau95b]) adapted for point clouds using the k-nearest neighbors instead of the one ring neighborhood. Another possibility is to use the weighted least squares projection ([HG01], [KG00]): the surface is iteratively projected onto the plane defined by the weighted barycenter o and the normal estimated using the weighted neighborhood covariance matrix. Weights are a simple gaussian ponderation on the distance to p and the size of the gaussian kernel is a parameter that controls the amount of smoothing. This projection process is in fact an order 1 projection motion (MLS1) that is analyzed in this chapter. To make the projection more efficient, [PKG06] proposed to subsample the point cloud. It yields a scale space decomposition where at each level the surface is smoothed and sub-sampled. The scale space decomposition is then applied to the multi-scale freeform deformation and to the morphing problem, with satisfactory results.

The moving least squares (MLS) were used to estimate curvatures. For example, in [YQ07], the authors use the MLS framework to build a closed form solution for curvature estimation. Indeed, surfaces implied by point clouds can be seen as the zero level set of an implicit function f. The gradient and Hessian Matrix of f is built. Finally, using formulas for the Gaussian and the mean curvature depending on the Hessian and gradient of f, those curvatures are computed.

In [CP03], the problem of estimating differential quantities on point clouds is

recast to that of fitting the local representation of the manifold by a jet. A jet is simply a truncated Taylor expansion. A n jet is a Taylor expansion truncated at order n. A jet of order n contains differential information up to the n-th order. In particular it is stated that a polynomial fitting of degree n estimates any $k^{th} order$ differential quantity to accuracy $O(h^{n-k+1})$. This implies that the coefficient of the first fundamental form and unit vector normal are estimated with $O(h^n)$ precision and the coefficients of the second fundamental form and shape operator are approximated with accuracy $O(h^{n-1})$, and so are the principal directions. In order to characterize curvature properties, the method resorts to the Weingarten map A of the surface, also called the shape operator, that is the tangent map of the Gauss map. Recall that the first and second fundamental forms I, II and A satisfy $II(\mathbf{t},\mathbf{t}) = I(A(\mathbf{t}),\mathbf{t})$ for any vector $\mathbf{t}$ of the tangent space. Second order derivatives are computed by building the Weingarten map of the osculating jet whose eigenvalues are the principal curvatures. Note that the described methods can be used either with a mesh or with a point cloud. Jets are in fact very related to MLS surfaces. Indeed, to estimate differential quantities a polynomial fitting of degree n is done, which is exactly what MLS does. Therefore the analyzis given in section 7.5 giving the equation governing MLS1 and MLS2 motions are valid for the jets too.

In this chapter, we give the exact partial differential equation that governs the MLS projection motion for order 1 and 2 MLS surfaces. We then link these PDEs to the surface curvatures.

Next section provides some tools to analyze numerically point cloud motions. The following analysis is in spirit very close to the image filter analysis performed in [BCM06].

7.2 Tools for numerical analysis of point cloud surface motions[1]

We always assume the existence of a smooth surface $\mathcal{M}$ supporting the point set. These surfaces are the boundaries of solid objects and can therefore be assumed to be locally Lipschitz graphs. However, for a mathematical analysis of smoothing algorithms and curvature estimations on the surface, we shall always assume that the surface is a C^∞ embedded manifold, known from its samples denoted by $\mathcal{M}_S$. This is not a limitation, in the sense that any finite sample set can be anyway interpolated by an arbitrarily smooth surface. Let $p(x_p, y_p, z_p)$ be a point of the surface $\mathcal{M}$. At each non umbilical point p, consider the principal curvatures k_1 and k_2 linked to the principal directions $\mathbf{t}_1$ and $\mathbf{t}_2$, with $k_1 > k_2$ where $\mathbf{t}_1$ and $\mathbf{t}_2$ are orthogonal vectors. (At umbilical points, any orthogonal pair $(\mathbf{t}_1, \mathbf{t}_2)$ can be taken.) Set $\mathbf{n} = \mathbf{t}_1 \times \mathbf{t}_2$ so

[1] Some theorems are in fact already in chapter 3, we rewrite them for clarity.

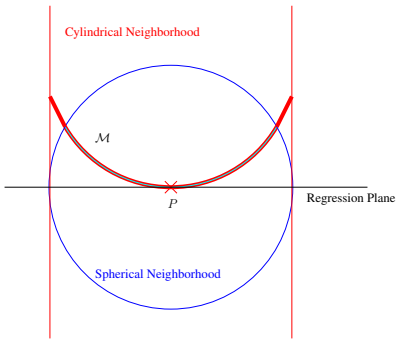


Figure 7.1: Comparison between cylindrical and spherical neighborhoods

that $(\mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$ is an orthonormal basis. The quadruplet $(p, \mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$ is called the local intrinsic coordinate system. In this system we can express locally the surface as a C^2 graph $z = f(x, y)$. By Taylor expansion,

$$z = f(x, y) = \frac{1}{2}(k_1 x^2 + k_2 y^2) + o(x^2 + y^2).^{2} \tag{7.1}$$

Notice that the sign of the pair (k_1, k_2) depends on the arbitrary surface orientation. Points where k_1 and k_2 have the same sign are called parabolic, and points where they have opposite signs are hyperbolic. Consider two kinds of neighborhoods in $\mathcal{M}$ for p defined in the local intrinsic coordinate system $(p, \mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$:

- the *spherical* neighborhood $\mathcal{B}_r = B_r(p) \cap \mathcal{M}$ is the set of all points m of $\mathcal{M}$ with coordinates (x, y, z) satisfying $(x - x_p)^2 + (y - y_p)^2 + (z - z_p)^2 < r^2$
- the *cylindrical neighborhood* $\mathcal{C}_r = C_r(p) \cap \mathcal{M}$ is the set of all points $m(x, y, z)$ on $\mathcal{M}$ such that $(x - x_p)^2 + (y - y_p)^2 < r^2$.

The spherical neighborhood in the sampled surface $\mathcal{M}_2$ is the only neighborhood to which there is a direct numerical access, and the discrete operators is defined with it. Nevertheless, for the forthcoming asymptotic numerical analysis, the cylindrical neighborhood will prove much handier than the spherical one. The next technical lemma justifies its use in theoretical calculations.

Lemma 3. *Integrating on $\mathcal{M}$ any function $f(x, y)$ such that $f(x, y) = O(r^n)$ on a cylindrical neighborhood $\mathcal{C}_r$ instead of a spherical neighborhood $\mathcal{B}_r$ introduces an $o(r^{n+4})$ error. More precisely:*

$$\int_{\mathcal{B}_r} f(x, y) dm = \int_{x^2+y^2<r^2} f(x, y) dx dy + O(r^{4+n}). \tag{7.2}$$

[2] We could use $z = f(x, y) = -\frac{1}{2}(k_1 x^2 + k_2 y^2) + o(x^2 + y^2)$ at the cost of changing the orientation and sign of k_1,k_2.

Proof. The surface area element of a point $m(x, y, z(x, y))$ on the surface $\mathcal{M}$, expressed as a function of x, y, dx and dy is $dm(x, y) = \sqrt{1 + z_x^2 + z_y^2}dxdy$. One has $z_x = k_1 x + O(r^2)$ and $z_y = k_2 y + O(r^2)$. Thus

$$dm(x, y) = \sqrt{(1 + k_1^2 x^2 + k_2^2 y^2 + O(r^3))}dxdy$$

which yields

$$dm(x, y) = (1 + O(r^2))dxdy. \tag{7.3}$$

Using (7.3), the integrals we are interested in become

$$\int_{\mathcal{C}_r} f(x, y)dm = (1 + O(r^2)) \int_{\mathcal{B}_r} f(x, y)dxdy; \tag{7.4}$$

$$\int_{\mathcal{B}_r} f(x, y)dm = (1 + O(r^2)) \int_{\mathcal{C}_r} f(x, y)dxdy \tag{7.5}$$

$$= (1 + O(r^2)) \int_{x^2+y^2<r^2,\ (x,y,z)\in\mathcal{M}} f(x, y)dxdy.$$

This last form is amenable to analytic computations. Consider polar coordinates (ρ, θ) such that $x = \rho\cos\theta$ and $y = \rho\sin\theta$ with $-r \leq \rho \leq r$ and $0 \leq \theta \leq \pi$. Then for $m(x, y, z)$ belonging to the surface $\mathcal{M}$, we have $z = \frac{1}{2}\rho^2(k_1\cos^2\theta + k_2\sin^2\theta) + O(r^3)$. Fixing θ we obtain $z = \frac{1}{2}\rho^2 k(\theta) + O(r^3)$, where $k(\theta) = k_1\cos^2\theta + k_2\sin^2\theta$. The condition that (x, y, z) belongs to the neighborhood $\mathcal{B}_r$ can therefore be rewritten as $\rho^2 + z^2 < r^2$, that is

$$\rho^2 + \frac{1}{4}k(\theta)^2\rho^4 < r^2 + O(r^5)$$

Computing the boundaries $\pm\rho(\theta)$ of this neighborhood yields $\rho(\theta)^2 + \frac{1}{4}k(\theta)^2\rho(\theta)^4 - r^2 + O(r^5) = 0$. Thus

$$\rho(\theta)^2 = \frac{-1 + \sqrt{1 + k(\theta)^2(r^2 + O(r^5))}}{\frac{1}{2}k(\theta)^2}.$$

This yields $\rho(\theta) = r - \frac{1}{2}k(\theta)^2 r^3 + O(r^4)$. We shall use this estimate for the error term E appearing in

$$\begin{aligned}\int_{\mathcal{B}_r} f(x, y)dxdy &= \int_{[0,2\pi]}\int_{[0,\rho(\theta)]} f(x, y)\rho d\rho d\theta \\ &= \int_{[0,2\pi]}\int_{[0,r]} f(x, y)\rho d\rho d\theta - E \\ &= \int_{\mathcal{C}_r} f(x, y)dxdy - E,\end{aligned}$$

with $E =: \int_{[0,2\pi]} \int_{[\rho(\theta),r]} f(x,y)\rho d\rho d\theta$. Thus

$$|E| \leq 2\pi \sup_{x^2+y^2\leq r^2} |f(x,y)| \max(k_1^2, k_2^2) r^4 + O(r^5).$$

In particular if $f(x,y) = O(r^n)$, then $|E| \leq O(r^{4+n})$. Finally we have

$$\int_{\mathcal{B}_r} f(x,y)dxdy = \int_{\mathcal{C}_r} f(x,y)dxdy + O(r^{4+n}). \tag{7.6}$$

Combining (7.4), (7.5) and (7.6) yields (7.2). □

This lemma will prove very useful for the rest of the chapter and in particular in the next section where analysis are given for various curvature estimates.

Lemma 3, as well as all theorems in the remainder of this chapter will assume that the surface is a uniform Lebesgue measure. A constant sampling density is therefore necessary. This constant density is approximated on discrete data by weighting each point by a weight inversely proportional to its initial density. More precisely, let p be a point and $\mathcal{N}_r(p) = \mathcal{M}_s \cap B_r(p)$. Each point q should ideally have a weight $0 \leq w(q) \leq 1$ such that $\sum_{q\in\mathcal{N}_r(p)} w(q) = 1$. This amounts to solve a huge linear system. For this reason, we shall be contented with ensuring $\sum_{q\in\mathcal{N}_r(p)} w(q) \simeq 1$ by taking $w(p) = \frac{1}{\sharp(B_p(r))}$, as proposed in [UH08].

7.3 Regression-free curvature estimates

This section finds the form of the differential operators underlying four different discrete schemes based on local cloud point statistics, and proposing discrete analogues of the "second fundamental forms" or of the "principal curvatures". These discrete schemes have very simple and robust form, being based on the computation of local moments and eigenvalues of the point cloud. We shall see that they actually compute nonlinear differential operators linked to the principal curvatures.

7.3.1 A discrete "second fundamental form" [BC94]

Let$(p_i)_{i\in 1\dots N}$ be the set of neighbors of a point p with normal $\mathbf{n}$. This paper proposes to build the "second fundamental form matrix" as follows. (Although this covariance matrix is not, as we shall see, consistent with the second fundamental form, it is thus called in this paper, and actually has, as we shall see, the principal directions as eigenvectors.) Let $s_i = (p_i - p)^T \cdot \mathbf{n}$, let $\mathbf{t}_1$, $\mathbf{t}_2$ be two orthonormal vectors of p's tangent plane, and

$$\alpha_i = s_i \cdot \begin{pmatrix} (p_i - p)\cdot \mathbf{t}_1 \\ (p_i - p)\cdot \mathbf{t}_2 \end{pmatrix} = ((p_i - p)^T \cdot \mathbf{n}) \cdot \begin{pmatrix} (p_i - p)\cdot \mathbf{t}_1 \\ (p_i - p)\cdot \mathbf{t}_2 \end{pmatrix}.$$

The α_i are projections of the vectors $(p_i - p)$ onto the tangent plane to p, weighted by their distance to this plane. The "second fundamental form matrix" is the covariance of these vectors, namely

$$\Sigma_d = \sum_{i=1}^{N} (\alpha_i - \alpha_m) \cdot (\alpha_i - \alpha_m)^T \tag{7.7}$$

where $\alpha_m = \frac{1}{N}\sum_{i=1}^{N} \alpha_i$ and in Σ_d the d stands for "discrete". To compute the underlying differential operators, two assumptions are made throughout this chapter. The first one is that the surface sampling is uniform with respect to the area measure on the surface. The second one is that this sampling is dense enough, so that the averages taken on neighborhoods can be interpreted as integrals. Under this interpretation, we can reinterpret the sum in (7.7) as an integral on a cylindrical neighborhood of p, assuming the data point set to be a locally smooth manifold. In the local intrinsic surface coordinate system at point p, $(p, \mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$, the surface can be written as a graph $z = \frac{1}{2}(k_1 x^2 + k_2 y^2) + o(r^2)$. Thus the vectors α_i are replaced by a continuous vector $\alpha(x, y)$ defined by

$$\alpha(x, y) = \frac{1}{2}(k_1 x^2 + k_2 y^2) \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{2} \cdot \begin{pmatrix} k_1 x^3 + k_2 y^2 x \\ k_1 x^2 y + k_2 y^3 \end{pmatrix} + o(r^3). \tag{7.8}$$

Under the interpretation taken above the "second fundamental matrix" rewrites

$$\Sigma = \int_{\mathcal{B}_r} (\alpha(x, y) - \alpha_m) \cdot (\alpha(x, y) - \alpha_m)^T dm(x, y) \tag{7.9}$$

where

$$\alpha_m = \frac{1}{\text{meas}(\mathcal{B}_r)} \int_{\mathcal{B}_r} \alpha(x, y) dm(x, y). \tag{7.10}$$

The proposition made in [BC94] is to extract the surface principal curvatures and their corresponding directions at p from this covariance matrix, as its eigenvalues and eigenvectors. The next theorem checks if this works asymptotically in the continuous model.

Theorem 5. *The eigenvectors of the "second fundamental form matrix" Σ give the principal directions with error $o(r^8)$. But the eigenvalues of Σ are not the principal curvatures as they satisfy*

$$\lambda_1 = \frac{1}{4}\frac{r^8}{8}\frac{\pi}{8}(5k_1^2 + 2k_1k_2 + k_2^2) + o(r^8) \text{ and } \lambda_2 = \frac{1}{4}\frac{r^8}{8}\frac{\pi}{8}(k_1^2 + 2k_1k_2 + 5k_2^2) + o(r^8)$$

where k_1 and k_2 are the principal curvatures at p

Proof. In the continuous model α_m therefore is close to zero because the integrated function is odd on a symmetric domain. More precisely, using Lemma 3 in (7.10) and writing $\alpha_m = (\alpha_{mx}, \alpha_{my})$,

$$\alpha_{mx} = \frac{1}{2\pi r^2} \int_{x^2+y^2<r^2} (k_1 x^3 + k_2 y^2 x_i + o(r^5)) dxdy = o(r^3)$$

and similarly

$$\alpha_{my} = o(r^3).$$

By Lemma 3 again, the covariance matrix (7.9) satisfies $\Sigma = \int_{x^2+y^2<r^2} \alpha(x,y) \cdot \alpha(x,y)^T dxdy + o(r^8)$, and, using (7.8), we can calculate its four components as follows.

$$\begin{aligned}
\Sigma_{11} &= \frac{1}{4} \int_{x^2+y^2<r^2} (k_1 x^3 + k_2 y^2 x)^2 dxdy + o(r^8) \\
&= \frac{1}{4} \int_{x^2+y^2<r^2} (k_1^2 x^6 + 2k_1 k_2 x^4 y^2 + k_2^2 y^4 x^2) dxdy + o(r^8) \\
&= \frac{1}{4} \int_{\theta\in[0,2\pi],\rho\in[0,r]} \rho^6 (k_1^2 cos(\theta)^6 + 2k_1 k_2 cos(\theta)^4 \sin(\theta)^2 + k_2^2 \cos(\theta)^2 \sin(\theta)^4) \rho d\rho d\theta + o(r^8) \\
&= \frac{1}{4}\frac{r^8}{8} \int_{\theta\in[0,2\pi]} (k_1^2 cos(\theta)^6 + 2k_1 k_2 cos(\theta)^4 \sin(\theta)^2 + k_2^2 \cos(\theta)^2 \sin(\theta)^4) d\theta + o(r^8) \\
&= \frac{1}{4}\frac{r^8}{8} (k_1^2 \frac{5\pi}{8} + 2k_1 k_2 \frac{\pi}{8} + k_2^2 \frac{\pi}{8}) + o(r^8) \\
&= \frac{1}{4}\frac{r^8}{8}\frac{\pi}{8} (5k_1^2 + 2k_1 k_2 + k_2^2) + o(r^8)
\end{aligned}$$

By exchanging the roles of k_1, k_2, and x, y respectively, we get

$$\Sigma_{22} = \frac{1}{4}\frac{r^8}{8}\frac{\pi}{8} (k_1^2 + 2k_1 k_2 + 5k_2^2) + o(r^8).$$

Σ being a symmetric matrix, $\Sigma_{12} = \Sigma_{21}$ and the integrated function being odd,

$$\begin{aligned}
\Sigma_{12} &= \frac{1}{4} \int_{x^2+y^2<r^2} (k_1 x^3 + k_2 y^2 x)(k_1 x^2 y + k_2 y^3) dxdy + o(r^8) \\
&= \frac{1}{4} \int_{x^2+y^2<r^2} (k_1^2 x^5 y + 2k_1 k_2 x^3 y^3 + k_2^2 y^5 x) dxdy + o(r^8) \\
&= o(r^8).
\end{aligned}$$

Thus, Σ is equivalent to a diagonal matrix whose principal directions are t_1 and t_2, which validates the theoretical requirements, t_1 and t_2 being the principal directions at point p. However, the corresponding eigenvalues are

$$\lambda_1 = \frac{1}{4}\frac{r^8}{8}\frac{\pi}{8}(5k_1^2 + 2k_1k_2 + k_2^2) + o(r^8)$$

$$\lambda_2 = \frac{1}{4}\frac{r^8}{8}\frac{\pi}{8}(k_1^2 + 2k_1k_2 + 5k_2^2) + o(r^8)$$

which are definitely different from $\lambda_1 = k_1$ and $\lambda_2 = k_2$. Only the absolute values of k_1 and k_2 can actually be deduced from Σ.

□

7.3.2 Another discrete "second fundamental form"

Another method is also introduced in [BC94] which, in a nutshell, computes the covariance matrix of the unit normal vectors projections onto the local tangent plane. By applying again the continuous asymptotic analysis of section 7.3.1, we shall see in Theorem 6 that this method actually computes discrete approximations of the squares of the principal curvatures. The discrete algorithm is as follows. Let $\mathcal{M}$ be a $\mathcal{C}^2$ surface and p be a point of $\mathcal{M}$. Let $(p_i)_i$ be the neighbors of p in a ball neighborhood of radius r. Denote by $\mathbf{n}_i$ the normal at p_i and define v_i as the projection of $\mathbf{n}_i$ onto the tangent plane at p, then the computed "curvatures" are defined as the eigenvalues of the covariance matrix of the vectors v_i. The vector v_i being the projection of $\mathbf{n}_i$ onto the tangent plane, we have:

$$v_i = \begin{pmatrix} \mathbf{n}_i \cdot \mathbf{t}_1 \\ \mathbf{n}_i \cdot \mathbf{t}_2 \end{pmatrix}$$

Set $v_m = \frac{1}{N}\sum_{i=1}^N v_i$. Then this new discrete covariance matrix writes $\Sigma_d = \sum_{i=1}^N (v_i - v_m) \cdot (v_i - v_m)^T$. In the continuous framework, the local points on the surface have coordinates $m(x,y) = (x, y, \frac{1}{2}(k_1x^2 + k_2y^2) + o(r^2))$ and the normal vector to this surface is $\frac{\partial m}{\partial x}(x,y) \wedge \frac{\partial m}{\partial y}(x,y) = (-k_1x, -k_2y, 1) + o(r)$. It follows that

$$v(x,y) = \frac{1}{\sqrt{1 + k_1^2x^2 + k_2^2y^2}} \begin{pmatrix} -k_1x \\ -k_2y \end{pmatrix} + o(r),$$

$$v_m = \frac{1}{\text{meas}(\mathcal{B}_r)} \int v(x,y) dm(x,y),$$

and the continuous covariance matrix is

$$\Sigma := \int_{\mathcal{B}_r} (v(x,y) - v_m) \cdot (v(x,y) - v_m)^T.$$

Theorem 6. *The eigenvalues of the covariance matrix Σ of the vectors $v(x,y)$ in the spherical neighorhood $\mathcal{B}_r$ are*

$$\frac{k_1^2 r^4 \pi}{4} + o(r^4) \text{ and } \frac{k_2^2 r^4 \pi}{4} + o(r^4).$$

Proof. Let us compute the mean v_m of $v(x,y)$ on the spherical neighborhood. By Lemma 3 the integral on a spherical neighborhood is asymptotically equivalent to the integral on a cylindrical neighborhood and more precisely,

$$\begin{aligned}
(\pi r^2) v_m \cdot t_1 &= \int_{x^2+y^2<r^2} \frac{-k_1 x}{\sqrt{1+k_1^2x^2+k_2^2y^2}} dxdy + o(r^3) \\
&= \int_{x^2+y^2<r^2} -k_1 x(1 - \frac{1}{2}k_1^2x^2 - \frac{1}{2}k_2^2y^2) dxdy + o(r^3) \\
&= -\int_{\theta\in[0,2\pi],\rho\in[0,r]} k_1\rho\cos(\theta)(1-\frac{\rho^2}{2}(k_1^2\cos^2\theta + k_2^2\sin^2\theta))\rho d\rho d\theta + o(r^3) \\
&= -\int_{\theta\in[0,2\pi],\rho\in[0,r]} k_1\rho^2\cos(\theta) + A\rho^4 d\rho d\theta + o(r^3) \text{ where A is a bounded constant} \\
&= -\frac{k_1}{3}\int_{\theta\in[0,2\pi]} r^3\cos(\theta) d\theta + o(r^3) \\
&= o(r^3)
\end{aligned}$$

and similarly

$$\begin{aligned}
(\pi r^2) v_m \cdot t_2 &= \int_{x^2+y^2<r^2} \frac{-k_2 y}{\sqrt{1+k_1^2x^2+k_2^2y^2}} dxdy + o(r^4) \\
&= o(r^3)
\end{aligned}$$

Thus the coefficients of Σ satisfy, again by Lemma 3,

$$\begin{aligned}
\Sigma_{11} &= \int_{x^2+y^2<r^2} \frac{k_1^2x^2}{1+k_1^2x^2+k_2^2y^2} dxdy + o(r^4) \\
&= \int_{x^2+y^2<r^2} k_1^2x^2(1-k_1^2x^2-k_2^2y^2) dxdy + o(r^4) \\
&= k_1^2 \int_{\theta\in[0,2\pi],\rho\in[0,r]} \rho^3\cos^2\theta(1-\rho^2(k_1^2\cos^2\theta + k_2^2\sin^2\theta)) d\rho d\theta + o(r^4) \\
&= \frac{k_1^2 r^4}{4}\int_{\theta\in[0,2\pi]} \cos^2\theta d\theta + r^6 A + o(r^4) \text{ with } A \text{ bounded} \\
&= \frac{k_1^2 r^4 \pi}{4} + o(r^4)
\end{aligned}$$

Similarly,

$$\begin{aligned}\Sigma_{22} &= \int_{x^2+y^2<r^2} \frac{k_2^2 y^2}{1+k_1^2x^2+k_2^2y^2} dxdy + o(r^4) \\ &= \frac{k_2^2 r^4 \pi}{4} + o(r^4)\end{aligned}$$

and

$$\begin{aligned}\Sigma_{12} = \Sigma_{21} &= \int_{x^2+y^2<r^2} \frac{k_1 k_2 xy}{1+k_1^2x^2+k_2^2y^2} dxdy + o(r^4) \\ &= \int_{x^2+y^2<r^2} k_1 k_2 xy(1-k_1^2x^2-k_2^2y^2)dxdy + o(r^4) \\ &= k_1k_2 \int_{\theta\in[0,2\pi],\rho\in[0,r]} \rho^3 \sin(\theta)\cos(\theta) - \rho^5 \sin(\theta)\cos(\theta)(k_1^2\cos^2\theta + k_2^2\sin^2\theta) d\rho d\theta + o(r^4) \\ &= \frac{k_1k_2}{4} \int_{\theta\in[0,2\pi]} \sin(\theta)\cos(\theta) r^4 d\theta + Ar^6 + o(r^4) \text{ where A is bounded} \\ &= r^4 \frac{k_1k_2}{4} \int_{\theta\in[0,2\pi]} \sin(\theta)\cos(\theta) d\theta + o(r^4) \\ &= o(r^4)\end{aligned}$$

□

Thus at order 4, Σ can be considered diagonal and its eigenvalues are Σ_{11} and Σ_{22} in the principal directions t_1 and t_2. They asymptotically give an approximation of each of the squared principal curvatures, but not their sign.

7.3.3 A third discrete "fundamental form"

The methods analyzed in sections 7.3.1 and 7.3.2 are akin to the original method introduced in [LT90]. Indeed, in [LT90] it was proposed to compute the covariance matrix of the normal vectors of the neighborhood (without projecting them in the local regression plane) and therefore get a 3×3 matrix instead of a 2×2 matrix. This is actually the simplest imaginable method and we shall see that it gives a result similar to section 7.3.2.

Theorem 7. *Let $\mathcal{M}$ be a $\mathcal{C}^2$ surface, let p be a point of $\mathcal{M}$. Then the three eigenvalues of the covariance matrix C of the unit normals in a neighborhood of radius r around p are asymptotically respectively equal to 1 and to the squares of the principal curvatures at p.*

Proof. A normal vector writes

$$N = \frac{1}{\sqrt{1 + k_1^2 x^2 + k_2^2 y^2}} \begin{pmatrix} -k_1 x \\ -k_2 y \\ 1 \end{pmatrix} + o(r).$$

As in the previous sections, we easily obtain by Lemma 3, $N_{mx} = o(r)$, $N_{my} = o(r)$, $N_{mz} = 1 + o(r)$. Thus again by Lemma 3,

$$C = \int_{x^2+y^2 \leq r^2} \frac{1}{1 + k_1^2 x^2 + k_2^2 y^2} \begin{pmatrix} k_1^2 x^2 & k_1 k_2 xy & k_1 x(1 - N_{mz}) \\ k_1 k_2 xy & k_2^2 y^2 & k_2 y(1 - N_{mz}) \\ k_1 x(1 - N_{mz}) & k_2 y(1 - N_{mz}) & (1 - N_{mz})^2 \end{pmatrix} dxdy + o(r^4)$$

and, by calculations exactly analogous to Section 7.3.2, $C_{11} = \frac{k_1^2 r^4 \pi}{4} + o(r^4)$, $C_{22} = \frac{k_2^2 r^4 \pi}{4} + o(r^4)$, $C_{12} = C_{21} = k_1 k_2 \int_{x,y} xydxdy = o(r^4)$, $C_{13} = C_{31} = C_{23} = C_{32} = C_{33} = o(r^4)$.

Thus the eigenvalues are asymptotically equal to $\frac{k_1^2 r^4 \pi}{4}$ and $\frac{k_2^2 r^4 \pi}{4}$, which also gives back the squares of the principal curvatures of the surface, but not their sign. □

7.3.4 A fourth discrete fundamental form: the surface variation

We shall now analyze a last variant introduced in [PGK02], the so called *surface variation*. It is again based on a local covariance analysis. Unlike the previous methods, the *surface variation* was not claimed to be a curvature estimate, but to be a measure of the neighborhood shape. This subsection establishes again a link between this discrete quantity and the principal curvatures of the surface.

Let p be a point with given neighborhood $\mathcal{B}_r$. Let o be the barycenter of the neighborhood. In $\mathbb{R}^3$, the coordinates are written with superscripts e.g. the coordinates of a point u are (u^1, u^2, u^3). Thus, for $i = 1, 2, 3$, $o^i = \frac{1}{\text{card}\,\mathcal{B}_r} \sum_{p_k \in \mathcal{B}_r} p_k^i$. The centered covariance matrix $\Sigma = (m_{ij})_{i,j=1,\cdots,3}$ is defined as $m_{ij} = \sum_{p_k \in \mathcal{B}_r} (p_k^i - o^i) \cdot (p_k^j - o^j)$ for $i, j = 1, 2, 3$. Let $\lambda_0 \leq \lambda_1 \leq \lambda_2$ be the eigenvalues of Σ with corresponding eigenvectors v_0, v_1, v_2. For $k = 0, \cdots, 2$,

$$\lambda_k = \sum_{p_i \in \mathcal{B}_r} \langle (p_i - o), v_k \rangle^2. \tag{7.11}$$

Each eigenvalue gives the variance of the point set in the direction of the corresponding eigenvector. Since v_1 and v_2 are the vectors that capture most variations, they define the PCA regression plane. The normal v_0 to this plane is the direction v minimizing $\sum_{p_i \in \mathcal{B}_r} \langle (p_i - o), v \rangle^2$. [PGK02] defines the surface variation by

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}. \tag{7.12}$$

This quantity measures the ratio of variance along the normal to the total variance. If the neighborhood is highly curved, its surface variation will be high and if the neighborhood is flat the surface variation will be small. This quantity has the property to be bounded between 0 (flat case) and $1/3$ (isotropic distribution case).

Lemma 4. *In the local intrinsic coordinate system, the barycenter of a neighborhood $\mathcal{B}_r$ of point p has coordinates $x_o = o(r^2)$, $y_o = o(r^2)$ and $z_o = \frac{Hr^2}{4} + o(r^2)$, where $H = \frac{k_1+k_2}{2}$ is the mean curvature at p.*

Proof. By Lemma 3 applied to the numerator and denominator of the following fraction, we have

$$\begin{aligned}
z_o = &\frac{\int_{\mathcal{B}_r} z dm}{\int_{\mathcal{B}_r} dm} = \frac{\int_{x^2+y^2<r^2} z(x,y)dxdy + O(r^5)}{\int_{x^2+y^2<r^2} dxdy + O(r^3)} \\
=&\frac{\int_{x^2+y^2<r^2} \left[\frac{1}{2}(k_1x^2 + k_2y^2) + o(x^2+y^2)\right] dxdy}{\int_{x^2+y^2<r^2} dxdy} + O(r^3) \\
=&\frac{1}{2\pi r^2}\int_{\rho=0}^{r}\int_{\theta=0}^{2\pi} \rho^2(k_1\cos^2\theta + k_2\sin^2\theta)\rho d\rho d\theta + o(r^2) \\
=&\frac{r^2}{8\pi}(k_1\pi + k_2\pi) + o(r^2) = \frac{Hr^2}{4} + o(r^2).
\end{aligned}$$

A similar but simpler computation yields the estimates of x_o and y_o. □

Theorem 8. *In the local coordinate system the surface variation σ satisfies*

$$\sigma = \frac{r^2}{16}\left(\frac{k_1^2+k_2^2}{2} - \frac{1}{3}k_1k_2\right) + o(r^2) \tag{7.13}$$

Proof. We need to explain what the covariance eigenvalues stand for. Each eigenvector v_i and associated eigenvalue λ_i represent a principal direction and the variation along this principal direction:

$$\lambda_i = \int_{m\in\mathcal{B}_r} \langle om, v_i\rangle^2 dm$$

Since we have $\lambda_0 \leq \lambda_1 \leq \lambda_2$, we can see that λ_0 is associated to the direction with the least variation namely the normal direction to the surface oz. Since the eigenvectors form an orthonormal basis, we have

$$\begin{aligned}
&\lambda_0 + \lambda_1 + \lambda_2 \\
&\quad= \int_{m\in\mathcal{C}_r} \langle om, v_0\rangle^2 + \langle om, v_1\rangle^2 + \langle om, v_2\rangle^2 dm \\
&\quad= \int_{m\in\mathcal{C}_r} \|om\|^2 dm
\end{aligned}$$

This yields

$$\lambda_0 + \lambda_1 + \lambda_2 = \int_{x^2+y^2<r^2} x^2 + y^2 + (z - z_o)^2 dxdy$$

and

$$\lambda_0 + \lambda_1 + \lambda_2 = \frac{\pi r^4}{2} + \lambda_0 + o(r^6) \tag{7.14}$$

We first compute λ_0, applying again Lemma 3 to get back to the easy cylindrical neighborhood.

$$\begin{aligned}
\lambda_0 &= \int_{x^2+y^2\leq r^2} (z - z_o)^2 dxdy \\
&= \int_{x^2+y^2\leq r^2} z^2 dxdy + z_o^2 \int_{x^2+y^2\leq r^2} dxdy - 2z_o \int_{x^2+y^2\leq r^2} z dxdy \\
&= \int_{x^2+y^2\leq r^2} z^2 dxdy + \frac{H^2 r^4}{16} * \pi r^2 - 2\frac{Hr^2}{4}\frac{r^4}{4}\pi H + o(r^6) \\
&= \frac{1}{4}(k_1^2 \int_{x^2+y^2\leq r^2} x^4 dxdy + k_2^2 \int_{x^2+y^2\leq r^2} y^4 dxdy + 2k_1k_2 \int_{x^2+y^2\leq r^2} x^2y^2 dxdy \\
&- \frac{H^2 r^6}{16}\pi + o(r^6) \\
&= \frac{1}{4}\frac{r^6}{6}(\frac{3\pi}{4}(k_1^2 + k_2^2) + k_1k_2\frac{\pi}{2}) - \frac{H^2 r^6}{16}\pi + o(r^6)
\end{aligned}$$

where $H = \frac{k_1+k_2}{2}$ is the mean curvature. Thus

$$\lambda_0 = \frac{\pi r^6}{32}(\frac{k_1^2 + k_2^2}{2} - \frac{1}{3}k_1k_2) + o(r^6) \tag{7.15}$$

Using 7.14 and 7.15 we get:

$$\sigma = \frac{\frac{r^2}{16}(\frac{k_1^2+k_2^2}{2} - \frac{1}{3}k_1k_2) + o(r^2)}{1 + \frac{r^2}{16}(\frac{k_1^2+k_2^2}{2} - \frac{1}{3}k_1k_2) + o(r^2)}$$

which finally yields:

$$\sigma = \frac{r^2}{16}\left(\frac{k_1^2 + k_2^2}{2} - \frac{1}{3}k_1k_2\right) + o(r^2)$$

□

The formula of the surface variation given by Theorem 8 indeed measures a sort of curvature. To interpret it we can notice that

- the surface variation is symmetric in k_1,k_2;
- in the case of a point lying on a sphere, $k_1 = k_2 = k$ so $\sigma_{sphere} = \frac{r^6}{24}k^2$;
- in the case of a saddle point $k_1 = k = -k_2$, $\sigma_{saddle} = \frac{r^6}{12}k^2$ so $\sigma_{sphere} < \sigma_{saddle}$;
- in the case of a cylinder $k_1 = k$, $k_2 = 0$, $\sigma_{cylinder} = \frac{r^6}{32}k^2$.

It follows from that the surface variation is not a discriminant information on the surface curvature, being unable to discriminate between very different local shapes.

7.4 The MLS1 projection

We shall now explore a more efficient way of computing locally at least the mean curvature, and to use it as a scale space. The main tool of the scale space proposed in [DMMSL09] is a simple projection of each surface point p on the local regression plane. This PCA regression plane is defined as the plane orthogonal to the least eigenvector of the centered local covariance matrix, and passing through the centroid of the neighborhood. The projection of p on this plane will be called p'. This projection method is the simplest instance of the *moving least square* method (MLS) by which each point of a surface is projected to a polynomial regression. The local barycenter can be considered as an MLS of order 0, MLS0, and the present projection on a plane is an order 1 MLS, which we shall denote by MLS1. The next lemma compares the normal to the PCA regression plane with the normal to the surface, $\mathbf{n}$ at p.

Lemma 5. *The normal* $\mathbf{v}$ *to the PCA regression plane in a spherical neighborhood* $\mathcal{B}_r$ *at* $p \in \mathcal{M}$ *is equal to the surface normal at point* p*, up to a negligible factor:* $\mathbf{v} = \mathbf{n} + O(r)$.

Proof. The local PCA regression plane of point p is characterized as the plane passing through the barycenter of the neighborhood $\mathcal{B}_r$ and with normal $\mathbf{v}$ minimizing:

$$I(\mathbf{v}) = \int_{\mathcal{B}_r} |\langle \mathbf{v}, pp' \rangle|^2 dp' \text{ s.t. } \|\mathbf{v}\| = 1$$

Denoting by (v_x, v_y, v_z) the coordinates of $\mathbf{v}$,

$$I(\mathbf{v}) = \int_{\mathcal{B}_r} (v_x x + v_y y + v_z \frac{1}{2}(k_1 x^2 + k_2 y^2) + o(r^2))^2 dxdy.$$

Considering the particular value $\mathbf{v} = (0, 0, 1)$ shows that the minimal value I_{min} of $I(\mathbf{v})$ satisfies $I_{min} \leq O(r^6)$. In consequence the minimum (v_x, v_y, v_z) satisfies $v_x \leq O(r)$ and $v_y \leq O(r)$. Thus, since $||\mathbf{v}|| = 1$, $v_z \geq 1 - O(r)$ and therefore $\mathbf{v} = \mathbf{n} + O(r)$. □

By Lemma 5, projecting p onto the regression plane induces a motion which is asymptotically in the normal direction: $p'p$ is almost parallel to $\mathbf{n}$. Lemma 4 stated that sending each point onto the barycenter of its neighborhood approximates the mean curvature motion. We shall discuss later on why this result cannot be used for implementing the mean curvature motion. Nevertheless, the simple projection of each surface point p onto its local regression plane approximates a 3D scale space (mean curvature motion) as shown in the next theorem, and this discrete approximation will show very efficient.

Theorem 9. *Let T_r be the operator defined on the surface $\mathcal{M}$ transforming each point p into its projection $p' = T_r(p)$ on the local regression plane. Then*

$$T_r(p) - p = \frac{Hr^2}{4}\mathbf{n} + o(r^2). \tag{7.16}$$

Proof. By Lemma 4 the barycenter o of $\mathcal{B}_r$ has local coordinates $\overrightarrow{po} = (o(r^2), o(r^2), \frac{Hr^2}{4} + o(r^2))$. On the other hand $\overrightarrow{pp'}$ is proportional to the normal to the regression plane, $\mathbf{v}$. Thus by Lemma 5 $\overrightarrow{pp'} = \lambda(O(r), O(r), 1 - O(r))$. To compute λ, we use the fact that p' is the projection on the regression plane of p, and that o belongs to this plane by definition. This implies that $\overrightarrow{pp'} \perp \overrightarrow{op'}$ and therefore

$$\lambda^2 O(r^2) + \lambda(1 - O(r))(H\frac{r^2}{4} + o(r^2) + \lambda(1 - O(r))) = 0,$$

which yields $\lambda = \frac{Hr^2}{4} + o(r^2)$ and therefore

$$\overrightarrow{pp'} = (O(r^3), O(r^3), \frac{Hr^2}{4} + o(r^2)) = \frac{Hr^2}{4}\mathbf{n} + o(r^2).$$

□

7.5 Asymptotics of MLS1 and MLS2

There is some particular interest in MLS1, because the recent meshing method uses it as a very simple and reversible smoothing tool for point clouds ([DMMSL09] and chapter 3). On the other hand many cloud point processing methods involve some variant of the MLS2 method to smooth, interpolate, or sub-sample a point cloud. MLS1 and MLS2 are smoothing operators and therefore could be used as *scale spaces*, that is, as iterative smoothing operators. But, following [DMMSL09] MLS1 indeed is a scale space and MLS2 is not, as illustrated in the experiments of Section 7.6. The theorems of this section clarify what happens, by first showing that

MLS1 implements very accurately a mean curvature motion, and second that MLS2 is insensitive to first, second, and third order intrinsic derivatives and has an order 4 difference to the original surface. Thus, it is to be expected that by MlS2 a smooth surface will have only a very small motion. This is verified in the experimental section. We start with the asymptotic analysis of MLS1.

Theorem 10. *Consider a smooth manifold and its intrinsic coordinates at a point $p(0,0)$, so that the Taylor expansion in a neighborhood of p satisfies*

$$z = f(x,y) = \frac{1}{2}k_1x^2 + \frac{1}{2}k_2y^2 + f_3(x,y) + f_4(x,y) + f_5(x,y) + O(r^6)$$

where f_i are homogeneous polynomials in x, y of global degree i. The order 1 MLS approximation $MLS1(p)$ of p in a radial neighborhood of p with radius r satisfies

$$< MLS1(p) - p, \mathbf{n} >= \frac{r^2}{8}(k_2 + k_1) + O(r^3)$$

where x and y are the coordinates, k_1 and k_2 are the principal curvatures and $\mathbf{n}$ is the normal to the surface at p, oriented towards the concavity.

Lemma 6. *One can choose the coordinates x and y in the regression plane at p so that, z being the coordinate in the direction of the normal plane, the equation of the manifold around p has the form $z = f(x,y) = \sum_{i,j=1}^{5} a_{ij}x^iy^j + o(|x^2+y^2|^3)$, and in addition $a_{ij} = \tilde{a}_{ij}(1+O(r))$, where $z = \tilde{f}(\tilde{x},\tilde{y}) = \sum_{i,j=1}^{5} a_{ij}\tilde{x}^i\tilde{y}^j + o(|\tilde{x}^2+\tilde{y}^2|^3)$ is the equation of the manifold in the coordinates $(\tilde{x},\tilde{y},\tilde{z})$ defined by the normal at p and the directions of the principal curvatures.*

Proof. Consider $(\tilde{x},\tilde{y},\tilde{z})$ the coordinates in the intrinsic frame such that $\tilde{x}$ and $\tilde{y}$ are the coordinates associated with the principal curvatures at p, and the plane $\tilde{x}p\tilde{y}$ is the tangent plane. Consider now coordinates (x,y,z) associated with the regression plane in a spherical neighborhood. Because the normal at the regression plane tends to the real normal when the spherical neighborhood shrinks, we can choose the coordinate axes (x,y) in the regression plane so that the rotation R which sends one frame to the other is close to the identity, namely

$$(\tilde{x},\tilde{y},\tilde{z}) = R(x,y,z) \tag{7.17}$$

is close to the identity: $R \to Id$ when $r \to 0$. More precisely, by Lemma 5, the normal $\mathbf{v}(r)$ to the PCA regression plane in a spherical neighborhood $\mathcal{B}_r$ at $p \in \mathcal{M}$ is equal to the surface normal at point p, up to a negligible factor: $\mathbf{v}(r) = \mathbf{n} + O(r)$. Thus we also have

$$R = R(r) = I + O(r). \tag{7.18}$$

Consider now the asymptotic expansion of $\tilde{z}$ as a function of $\tilde{x}, \tilde{y}$. (We assume the manifold to be at least C^5):

$$\tilde{z} - \tilde{g}(\tilde{x}, \tilde{y}) - O((\tilde{x}^2 + \tilde{y}^2)^{\frac{5}{2}}) = 0.$$

Because of the relation (7.17) we can consider the above equation as an implicit equation in z, x, y, R, namely

$$Q(x, y, z, R) - O((x^2 + y^2 + z^2)^{\frac{5}{2}}) = 0. \tag{7.19}$$

However, by the chain rule we have $\frac{\partial Q}{\partial z}(0, 0, 0, Id) = 1$. Thus by the implicit function theorem, there is a function h of class C^5 such that in a neighborhood of $(0, 0, 0, Id)$, (7.19) is equivalent to

$$z = h(x, y, R).$$

Since h is C^5 we can make a Taylor expansion and therefore get

$$z = g(x, y, R) + O(||R - Id||^5 + (x^2 + y^2)^{\frac{5}{2}}).$$

In particular for $R = Id$ we obtain by identification of the terms with degree lower than 5 that $g(x, y, Id) = \tilde{f}(x, y)$. Thus, all monomials $a_{ij}(R)x^i y^j$ in the expansion of f with respect to x, y satisfy $a_{i,j}(R) = \tilde{a}_{i,j}(Id) + O(I - R)$, which by (7.18) yields $a_{i,j}(r) = \tilde{a}_{i,j} + O(r)$. □

Proof of Theorem 10 In the local coordinate system, the surface can be described as the graph

$$z = f(x, y) = f_1(x, y) + f_2(x, y) + f_3(x, y) + f_4(x, y) + f_5(x, y) + o(|x^2 + y^2|^{5/2})$$

where

$$f_1(x, y) = a_{10}x + a_{01}y, \quad f_2(x, y) = a_{20}x^2 + a_{11}xy + a_{02}y^2$$
$$f_3(x, y) = a_{30}x^3 + a_{21}x^2y + a_{12}xy^2 + a_{03}y^3,$$
$$f_4(x, y) = a_{40}x^4 + a_{31}x^3y + a_{22}x^2y^2 + a_{13}xy^3 + a_{04}y^4;$$
$$f_5(x, y) = a_{50}x^5 + a_{41}x^4y + a_{32}x^3y^2 + a_{23}x^2y^3 + a_{14}xy^4 + a_{05}y^5.$$

Let us find the order 1 polynomial g that best fits this surface in the least squares sense. Set $g(x, y) = \alpha x + \beta y + \theta$ Then we must find the parameters $\Theta = \begin{pmatrix} \alpha & \beta & \theta \end{pmatrix}$ which minimize

$$\int_{x^2+y^2<r^2} (g(x, y) - f(x, y))^2 dxdy = \int_{x^2+y^2<r^2} (X\Theta^T - f(x, y))^2 dxdy,$$

where $X = \begin{pmatrix} x & y & 1 \end{pmatrix}$. This is a quadratic minimization and the derivation yields

$$\int_{x^2+y^2<r^2} X^T(X\Theta^T - f(x,y))dxdy = 0,$$

and therefore

$$\Theta^T = \left(\int_{x^2+y^2<r^2} (X^T X) \right)^{-1} \int_{x^2+y^2<r^2} (X^T f(x,y)).$$

Let us compute the matrix $M = \int_{x^2+y^2<r^2}(X^T X)$.

$$M = \int_{x^2+y^2<r^2} \begin{pmatrix} x^2 & xy & x \\ xy & y^2 & y \\ x & y & 1 \end{pmatrix} dxdy = \frac{\pi r^4}{4} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{4}{r^2} \end{pmatrix};$$

$$M^{-1} = \frac{4}{\pi r^4} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{r^2}{4} \end{pmatrix}.$$

We also have

$$X^T f_1(x,y) = \frac{\pi r^4}{4} \begin{pmatrix} a_{10} \\ a_{01} \\ 0 \end{pmatrix}, \quad X^T f_2(x,y) = \frac{\pi r^4}{4} \begin{pmatrix} 0 \\ 0 \\ a_{20}+a_{02} \end{pmatrix}$$

$$X^T f_3(x,y) = \frac{\pi r^4}{4} \begin{pmatrix} \frac{r^2}{6}(3a_{30}+a_{12}) \\ \frac{r^2}{6}(a_{21}+2a_{03}) \\ 0 \end{pmatrix}, X^T f_4(x,y) = \frac{\pi r^4}{4} \begin{pmatrix} 0 \\ 0 \\ \frac{r^2}{6}(3a_{40}+a_{22}+3a_{04}) \end{pmatrix},$$

$$X^T f_5(x,y) = \frac{\pi r^4}{4} \begin{pmatrix} \frac{r^4}{16}(5a_{50}+a_{32}+a_{14}) \\ \frac{r^4}{16}(a_{41}+a_{23}+5a_{50}) \\ 0 \end{pmatrix}$$

Thus the parameter Θ satisfies

$$\Theta = M^{-1}X^T \int_{x^2+y^2<r^2} f(x,y) = \begin{pmatrix} a_{10} + \frac{r^2}{6}(a_{12}+3a_{30}) + \frac{r^4}{16}(5a_{50}+a_{32}+a_{14}) + O(r^5) \\ a_{01} + \frac{r^2}{6}(3a_{03}+a_{21}) + \frac{r^4}{16}(a_{41}+a_{23}+5a_{05}) + O(r^5) \\ (a_{20}+a_{02})\frac{r^2}{4} + \frac{r^4}{24}(3a_{04}+a_{22}+3a_{40}) + O(r^6) \end{pmatrix}.$$

This means that the projection on the plane generates a motion of amplitude

$$\frac{r^2}{4}(a_{20}+a_{02}) + \frac{r^4}{24}(3a_{04}+a_{22}+3a_{40}) + O(r^6),$$

Finally, Lemma 6 permits to replace a_{ij} by $\tilde{a}_{ij}(1+O(r))$ where $\tilde{a}_{ij}$ are the coefficients in the frame defined by the principal curvature eigenvectors and the normal.

In this frame $\tilde{a}_{20} = \frac{1}{2}k_1$ and $\tilde{a}_{02} = \frac{1}{2}k_2$. So that the final formula for the projection on the regression plane is:

$$(\frac{r^2}{8}(k_2+k_1)+\frac{r^4}{24}(3\tilde{a}_{04}+\tilde{a}_{22}+3\tilde{a}_{40})+O(r^6))(1+O(r)) = \frac{r^2}{8}(k_2+k_1)+O(r^3)$$

□

7.5.1 The asymptotic behavior of MLS2

We need to specify a simple version of MLS2, prone to an asymptotic analysis, but in agreement with the literature. The following version seems to be a sort of common denominator of all versions. The first difficulty of MLS2 is the fact that a first reference frame must be found, and that then the mean square approximation by order 2 polynomials is made in this reference frame. The most natural frame is found by applying MLS1, and the coordinates (x, y) is therefore the coordinates in the regression plane in a spherical neighborhood $\mathcal{B}_r$. The second step is to find the mean square closest order 2 polynomial in the spherical neighborhood. Because of Lemma 3 we can specify without loss of generality or precision that the minimization is made in the cylindrical neighborhood $\mathcal{C}_r$. In that way, all integrals computed in the approximation process are integrals on the disk $x^2+y^2 \leq r^2$, which is numerically and formally convenient. Thus the MLS2 algorithm specified for the analysis works in the two steps:

1. compute the regression plane of the manifold in the spherical neighborhood $\mathcal{B}_r = B_r(p) \cap \mathcal{M}$;

2. call (x, y) the reference coordinates in the regression plane. Consider the restriction of the smooth manifold to the disk $D_r := x^2 + y^2 \leq r^2$, $z = f(x, y)$. Then find the order 2 polynomial $g(x, y)$ that best approximates f for the $L^2(D_r)$ distance;

3. set (in the reference frame) $MLS2(p) := (0, 0, g(0, 0))$.

The next theorem shows that unlike MLS1 which reveals the mean curvature, the difference between a point smoothed by MLS2 and its original position is very small (of order 4) and actually only reveals a fourth order intrinsic operator. The result is actually interesting because the revealed fourth order operator is a kind of bilaplacian. The evolution by MLS2 is a fourth order equation that is intuitively well-posed at least for short times. Viewed that way, the next theorem finds a very simple and efficient numerical scheme for at least one fourth order intrinsic evolution.

Theorem 11. *Consider a smooth manifold and its intrinsic coordinates at a point $p(0,0)$, so that the Taylor expansion in a neighborhood of p satisfies*

$$z = f(x,y) = \frac{1}{2}k_1x^2 + \frac{1}{2}k_2y^2 + f_3(x,y) + f_4(x,y) + f_5(x,y) + O(r^6)$$

where f_i are homogeneous polynomials in x, y of global degree i. The order 2 MLS approximation $MLS2(p)$ of p in a cylindrical neighborhood of p with radius r satisfies

$$< MLS2(p) - p, \mathbf{n} >= -\frac{r^4}{48}(3a_{04} + a_{22} + 3a_{40})) + O(r^5)$$

where x and y are the coordinates of the principal curvatures, $a_{40} = \frac{1}{4!}\frac{\partial^4 f}{\partial x^4}$, $a_{04} = \frac{1}{4!}\frac{\partial^4 f}{\partial y^4}$, $a_{22} = \frac{1}{4!}\frac{\partial^4 f}{\partial x^2 \partial^2 y}$ are the fourth derivatives of the intrinsic equation at p in the directions of x, y and x, y respectively, and $\mathbf{n}$ is the normal to the surface at p, oriented towards the concavity.

Proof. Let us write $f(x,y) = f_1(x,y) + f_2(x,y) + f_3(x,y) + f_4(x,y) + f_5(x,y) + o(|x^2+y^2|^{5/2})$ where

$$f_1(x,y) = a_{10}x + a_{01}y, \quad f_2(x,y) = a_{20}x^2 + a_{11}xy + a_{02}y^2$$

$$f_3(x,y) = a_{30}x^3 + a_{21}x^2y + a_{12}xy^2 + a_{03}y^3,$$
$$f_4(x,y) = a_{40}x^4 + a_{31}x^3y + a_{22}x^2y^2 + a_{13}xy^3 + a_{04}y^4,$$
$$f_5(x,y) = a_{50}x^5 + a_{41}x^4y + a_{32}x^3y^2 + a_{23}x^2y^3 + a_{14}xy^4 + a_{05}y^5.$$

We look for the order 2 polynomial g that best fits this surface in the least squares sense,

$$g(x,y) = \alpha x^2 + \beta y^2 + \gamma xy + \delta x + \varepsilon y + \theta.$$

We therefore must find the parameters $\Theta = \begin{pmatrix}\alpha & \beta & \gamma & \delta & \varepsilon & \theta\end{pmatrix}$ which minimize

$$\int_{x^2+y^2<r^2} (g(x,y) - f(x,y))^2 dxdy = \int_{x^2+y^2<r^2} (X\Theta^T - f(x,y))^2 dxdy$$

where $X = \begin{pmatrix}x^2 & y^2 & xy & x & y & 1\end{pmatrix}$. This is a quadratic minimization and differentiating this integral with respect to Θ yields

$$\int_{x^2+y^2<r^2} X^T(X\Theta^T - f(x,y))dxdy = 0.$$

Writing $M = \int_{x^2+y^2<r^2} X^TX$, finally:

$$\Theta^T = \left(\int_{x^2+y^2<r^2} (X^TX)\right)^{-1} \int_{x^2+y^2<r^2} (X^Tf(x,y));$$

$$\Theta^T = M^{-1} \int_{x^2+y^2<r^2} X^T(f_1(x,y)+f_2(x,y)+f_3(x,y)+f_4(x,y)+f_5(x,y)+O((x^2+y^2)^3);$$

$$X^T X = \begin{pmatrix} x^4 & x^2y^2 & x^3y & x^3 & x^2y & x^2 \\ x^2y^2 & y^4 & xy^3 & xy^2 & y^3 & y^2 \\ x^3y & xy^3 & x^2y^2 & x^2y & xy^2 & xy \\ x^3 & xy^2 & x^2y & x^2 & xy & x \\ x^2y & y^3 & xy^2 & xy & y^2 & y \\ x^2 & y^2 & xy & x & y & 1 \end{pmatrix}.$$

When integrating on the disk, most terms vanish and we get

$$M = \frac{\pi r^4}{4} \begin{pmatrix} \frac{r^2}{2} & \frac{r^2}{6} & 0 & 0 & 0 & 1 \\ \frac{r^2}{6} & \frac{r^2}{2} & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{r^2}{6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & \frac{4}{r^2} \end{pmatrix};$$

$$M^{-1} = \frac{4}{\pi r^4} \begin{pmatrix} \frac{9}{2r^2} & \frac{3}{2r^2} & 0 & 0 & 0 & -\frac{3}{2} \\ \frac{3}{2r^2} & \frac{9}{2r^2} & 0 & 0 & 0 & -\frac{3}{2} \\ 0 & 0 & \frac{6}{r^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ -\frac{3}{2} & -\frac{3}{2} & 0 & 0 & 0 & r^2 \end{pmatrix}.$$

Therefore

$$\begin{aligned} \Theta^T =& M^{-1} \int_{x^2+y^2<r^2} X^T f_1(x,y) + M^{-1} \int_{x^2+y^2<r^2} X^T f_2(x,y) \\ &+ M^{-1} \int_{x^2+y^2<r^2} X^T f_3(x,y) + M^{-1} \int_{x^2+y^2<r^2} X^T f_4(x,y) \\ &+ M^{-1} \int_{x^2+y^2<r^2} X^T f_5(x,y) + + M^{-1} \int_{x^2+y^2<r^2} X^T O((x^2+y^2)^4) \end{aligned}$$

When computing $\int_{x^2+y^2<r^2} X^T f_1(x,y)$ most terms vanish, leaving

$$\int_{x^2+y^2<r^2} X^T f_1(x,y) = \frac{\pi r^4}{4} \begin{pmatrix} 0 \\ 0 \\ 0 \\ a_{10} \\ a_{01} \end{pmatrix}.$$

Again computing $\int_{x^2+y^2<r^2} X^T f_2(x,y)$ most terms vanish and we get

$$\int_{x^2+y^2<r^2} X^T f_2(x,y) = \frac{\pi r^4}{4} \begin{pmatrix} \frac{r^2}{6}(3a_{20}+a_{02}) \\ \frac{r^2}{6}(a_{20}+3a_{02}) \\ \frac{r^2}{6}a_{11} \\ 0 \\ 0 \\ a_{20}+a_{02} \end{pmatrix}.$$

Similarly,

$$\int_{x^2+y^2<r^2} X^T f_3(x,y) = \frac{\pi r^4}{4} \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{r^2}{6}(3a_{30}+a_{12}) \\ \frac{r^2}{6}(a_{21}+3a_{03}) \\ 0 \end{pmatrix};$$

$$\int_{x^2+y^2<r^2} X^T f_4(x,y) = \frac{\pi r^4}{4} \begin{pmatrix} \frac{r^4}{16}(5a_{40}+a_{22}+a_{04}) \\ \frac{r^4}{16}(a_{40}+a_{22}+5a_{04}) \\ \frac{r^4}{16}(a_{31}+a_{13}) \\ 0 \\ 0 \\ \frac{r^2}{6}(3a_{40}+a_{22}+3a_{04}) \end{pmatrix}.$$

Finally,

$$\int_{x^2+y^2<r^2} X^T(f_5(x,y)) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \int a_{50}x^5 + a_{32}x^4y^2 + a_{14}x^2y^4 dxdy \\ \int a_{41}x^4y^2 + a_{23}x^2y^4 + a_{05}y^6 dxdy \\ 0 \end{pmatrix}$$

$$= \frac{\pi r^4}{4} \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{r^4}{16}(5a_{50}+a_{32}+a_{14}) \\ \frac{r^4}{16}(a_{41}+a_{23}+5a_{05}) \\ 0 \end{pmatrix};$$

$$\int_{x^2+y^2<r^2} X^T(x^2+y^2)^3 = \frac{\pi r^4}{4}\begin{pmatrix} \frac{2r^6}{5} \\ \frac{2r^6}{5} \\ 0 \\ 0 \\ 0 \\ \frac{r^4}{2} \end{pmatrix}.$$

Multiplying all of these results by the matrix M^{-1}, we get

$$M^{-1}\int_{x^2+y^2<r^2} X^T f_1(x,y) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ a_{10} \\ a_{01} \\ 0 \end{pmatrix}; \tag{7.20}$$

$$M^{-1}\int_{x^2+y^2<r^2} X^T f_2(x,y) = \begin{pmatrix} a_{20} \\ a_{02} \\ a_{11} \\ 0 \\ 0 \\ 0 \end{pmatrix}; \tag{7.21}$$

$$M^{-1}\int_{x^2+y^2<r^2} X^T(f_3(x,y)) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{r^2}{6}(3a_{30}+a_{12}) \\ \frac{r^2}{6}(a_{21}+3a_{03}) \\ 0 \end{pmatrix}; \tag{7.22}$$

$$M^{-1}\int_{x^2+y^2<r^2} X^T(f_4(x,y)) = \begin{pmatrix} \frac{r^2}{8}(6a_{40}+a_{22}) \\ \frac{r^2}{8}(a_{22}+6a_{04}) \\ \frac{3r^2}{8}(a_{31}+a_{13}) \\ 0 \\ 0 \\ -\frac{r^4}{48}(3a_{40}+a_{22}+3a_{04}) \end{pmatrix}; \tag{7.23}$$

$$M^{-1}\int_{x^2+y^2<r^2} X^T(f_5(x,y)) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{r^4}{16}(5a_{50}+a_{32}+a_{14}) \\ \frac{r^4}{16}(a_{41}+a_{23}+5a_{05}) \\ 0 \end{pmatrix}; \tag{7.24}$$

$$M^{-1}\int_{x^2+y^2<r^2} X^T(x^2+y^2)^3 = \begin{pmatrix} \frac{33r^4}{20} \\ \frac{33r^4}{20} \\ 0 \\ 0 \\ 0 \\ -\frac{7r^6}{10} \end{pmatrix}, \tag{7.25}$$

and combining equations (7.20), (7.21), (7.22), (7.23), (7.24) and (7.25) we finally obtain the parameter Θ

$$\Theta^T = \begin{pmatrix} a_{20} + \frac{r^2}{8}(a_{22}+6a_{40}) + O(r^4) \\ a_{02} + \frac{r^2}{8}(a_{22}+6a_{04}) + O(r^4) \\ a_{11} + \frac{3r^2}{8}(a_{13}+a_{31}) + O(r^4) \\ a_{10} + \frac{r^2}{6}(a_{12}+3a_{30}) + \frac{r^4}{16}(5a_{50}+a_{32}+a_{14}) + O(r^5) \\ a_{01} + \frac{r^2}{6}(3a_{03}+a_{21}) + \frac{r^4}{16}(a_{41}+a_{23}+5a_{05}) + O(r^5) \\ -\frac{r^4}{48}(3a_{04}+a_{22}+3a_{40}) + O(r^6) \end{pmatrix}$$

so that the MLS2 projection satisfies

$$g(0,0) = -\frac{r^4}{48}(3a_{04}+a_{22}+3a_{40}) + O(r^6).$$

Finally Lemma 6 permits to replace

$$g(0,0) = -\frac{r^4}{48}(3a_{04}+a_{22}+3a_{40}) + O(r^6).$$

by

$$-(\frac{r^4}{48}(3\tilde{a}_{04}+\tilde{a}_{22}+3\tilde{a}_{40}) + O(r^6))(1+O(r)) = -\frac{r^4}{48}(3\tilde{a}_{04}+\tilde{a}_{22}+3\tilde{a}_{40}) + O(r^5)$$

□

7.6 Numerical experiments

This section shows experiments with the most significant algorithms described in the previous sections. A simulated randomly sampled sphere plays the role of numerical pattern. In particular we evaluate the curvatures given by MLS1 on the sphere, and the curvatures given by MLS2 projection, by polynomial regression. We also compute the curvature estimated by the method described in [BC94] and by the surface variation of [PGK02]. The results are compared by giving the mean

estimated curvature and its standard variation. The input data is a randomly sampled sphere with radius 2 corrupted with added centered Gaussian noise of variance 0.1.

Iteration	MLS1		MLS2	
	mean	standard variation	mean	standard variation
0	0.5828	2.8609	0.052	1.2879
1	0.5158	1.2434	0.4920	1.0053
2	0.5079	0.3196	0.5083	0.1259
3	0.5102	0.0253	0.5073	0.1001
4	0.5136	0.0189	0.5068	0.0855
5	0.5171	0.0165	0.5065	0.0749
10	0.5356	0.0156	0.5058	0.0489

Figure 7.2: Comparison of the curvature estimation by iteration of the MLS1 projection and iterations of the MLS2 projection

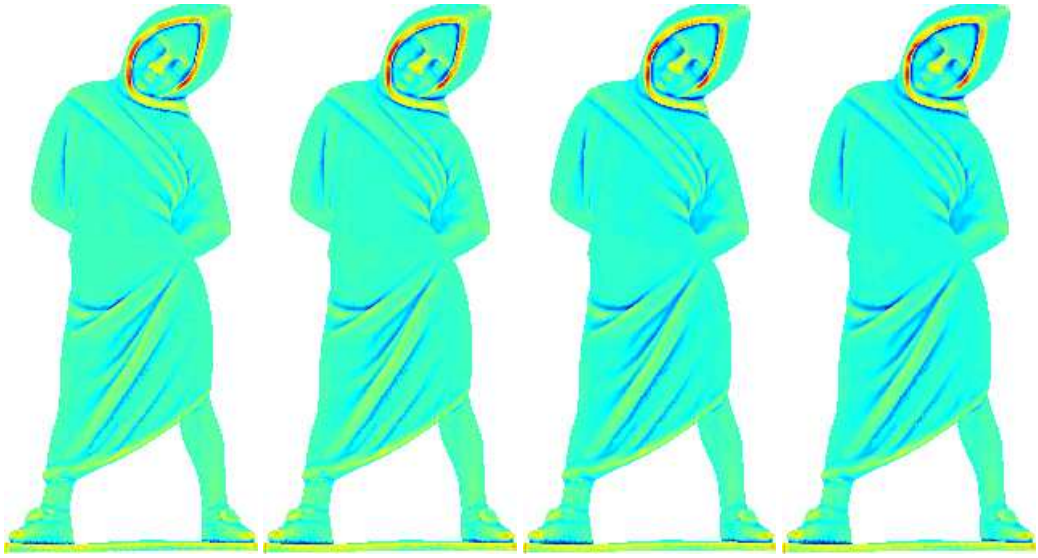
By comparing the values in tab. 7.2, two conclusions can be drawn: first, MLS1 is much more stable than the MLS2 projection, which can be observed by the standard variation on the estimates. Second, the MLS2 projection does not implement a mean curvature motion. The MLS1 projection instead yields an increase of the mean curvature (i.e., the sphere radius decreases, which is expected from a mean curvature motion). The standard deviation of the radius of the sphere gets smaller by MLS1 than by MLS2, meaning that MLS1 is a better curvature estimator. The surface variation of this point set is 0.1419 with standard variation 0.009. Finally, the mean curvature computed by normal covariance analysis is 2.4565 with variance 0.1026.

On real surfaces (Figs 7.3, 7.4 and 7.5), we present various curvature distribution and surface variation distributions. Intuitively, a mean curvature motion corresponds well to the MLS1 projection.

To better judge of the smoothing effect of the projection on the regression plane operator, we show the following experiment. First a consistently oriented point set is built (see [DMMSL09] or chapter 3 for an efficient way of doing so). This normal orientation yields the sign of the mean curvature, by computing the scalar product of the oriented normal and the displacement vector. Each point is then plotted in a different color according to its sign, blue for positive and red for negative (see Fig 7.6). This experiment shows that, at the beginning, the sign captures noise and small textures (i.e., small scale variations), and after some iterations, the shape is smoothed and the sign captures the geometry of the shape (large scale variations).

(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4

Figure 7.3: Curvature evolution by iterative projection on MLS1



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4

Figure 7.4: Curvature evolution by iterative projection on MLS2 .

Conclusion

In this chapter, we analyzed several proposition of discrete operators computing curvatures or curvature-like operators computed on surfaces defined by raw point

(a) Surface variation (b) Curvature by normal covariance analysis

Figure 7.5: Other curvature estimates .

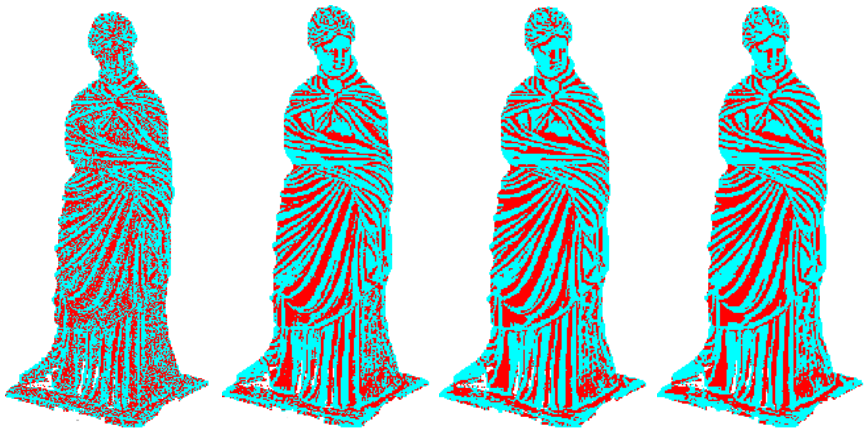


Figure 7.6: Evolution of the motion direction with projection iterations.

clouds. We showed that these clever methods unfortunately only at best recover the squared principal curvatures, and loose their signs. The analysis of MLS2 led us to a new intrinsic partial differential equation whose theoretical analysis could prove interesting. Computing k_1 and k_2 with their signs by a local statistical filter remains an open problem. Another problem arising from the present study can be called the *constant sampling density problem*: starting from an irregular sampling, is it possible to compute weights associated with each sample, in such a way that the

sum of weights is constant over all discrete Euclidean neighborhoods with a fixed size? To the best of our knowledge, there is no answer to this question yet, although this problem appears crucial to the field of numerical surface motion. Finally the simplest analysis performed in this chapter (the analysis of the MLS1 projection) yields a mean curvature motion, and this operator, proven very robust to irregular sampling, has already multiple practical applications (see chapters 3 and 4 which are detailed versions of [DMMSL09] and [DMAL10]).

Chapter 8

Color Cloud Visualization

Contents

Abstract: This chapter shows a striking illustration of the robustness of the scale space meshing method. The authors of [BLM10] have recently shown that the color histograms of natural images are essentially two-dimensional. Their main tool to prove this fact was to apply the projection on regression plane filter to the color histogram, a 3D point cloud, and to show that after a few iterations the point cloud stabilizes on a 2D surface. This occurs in general without any visible alteration of the image when replacing the original color by the filtered colors. But to do so, the projection filter must be applied at a quite different scale than for high quality point clouds. The numerical situation is much more involved, the size of the neighborhoods for the projection fitter being about 10% the point cloud diameter. In this chapter, the scale space meshing method is optimized to handle such very large neighborhoods, which raises serious memory management issues. While in [BLM10] the point clouds are only filtered by the projection filter, here we show that they can also be meshed. This reveals better the fascinating topology of color point clouds and the improved implementation permits to handle very large color images, confirming and revealing their 2D structure. Processing the color coulds (which have a very low signal noise ratio), proves that scale space meshing can not only handle high precision but also noisy data.

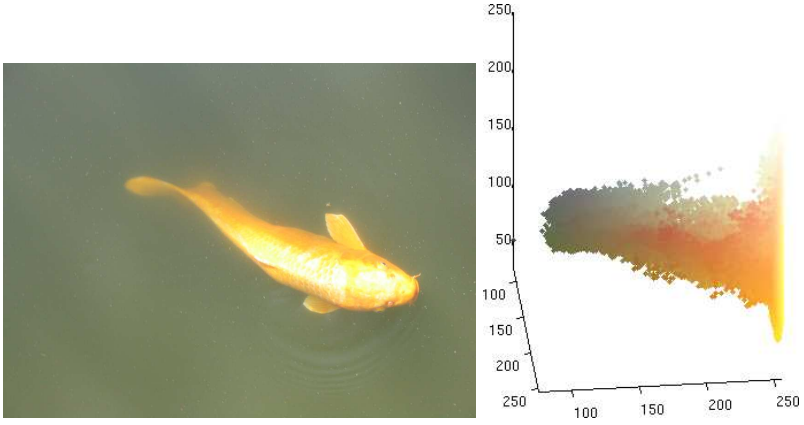
Introduction

Color images are functions defined on $\Omega \subset \mathbb{R}^2$ with values commonly clipped in $[0, 255]^3$ in the RGB color space. By dissociating pixel values from pixel positions, we get a set of 3D values in the color cube $[0, 255]^3$. These values constitute the color histogram of the image, a "color cloud" that will be filtered by the algorithm. The idea behind the projection on the regression plane is that the underlying structure of the color cloud is essentially 2-dimensional, otherwise the whole procedure would be pointless. The question of the dimensionality of color clouds has been investigated: [OW04] proposed a linear model to account for the color clouds structures. In [BLM10], it is shown that the 2D model accounted much better for the colors in the image. By a statistical analysis and numerical comparison between a 2D filtering and 1D filtering, this paper confirms that the color clouds are essentially 2D, roughly up to a 8% error. In other terms the clouds can be modeled as a (thick) 2D manifold. By the projection filter, the color clouds can be filtered to become really 2D surfaces, whose visualization is no more obscured by a few stray color points. Here, thanks to a careful numerical analysis, we were able to test this 2D hypothesis on far larger color images (2048×1536 or 3648×2736 pixels) than those

studied in [BLM10]. In addition, the filtered color clouds could then be meshed by scale space meshing to get a visualization of the color sheets at any smoothing scale.

8.1 Color clouds

Each pixel stores a five dimensional information (2 parameters for space information and 3 parameter for color information: red, green and blue channel). As stated before, if we consider only the color information, then we have a 3D point cloud contained in the $([0, 255]^3)$ cube. An example of a natural image and its color cloud can be seen on figure 8.1.



(a) Fish image (b) Fish color cloud

Figure 8.1: Example of a color cloud .

8.2 Implementation

The main difference with usual point cloud surfaces coming from triangulation scanners is that such data bring us close to the complexity worst case. Indeed the points are much more spread out in the color cube than they are in a good quality scan. This means that many more octree cells are actually occupied. As a result an acceleration of the code must be introduced. A difference is that the octree size stays always the same: $S = 255$ (in fact we set it to 265 to allow for a small error). Given a filtering radius r specified by the user (which according to [BLM10] goes from 5, 10 to 20), an octree depth d can be inferred so that the leaf size l is:

$$\frac{S}{2^{l+2}} < r < \frac{S}{2^{l+1}}$$

Because of the order of magnitude of the radius, this would mean an octree depth of less than 4, and would therefore make the neighbor search intractable. In order to accelerate the search, we use the fact that we do not need the neighbors individually. Indeed, we only need to compute their covariance matrix and barycenter. If each cell (be it leaf or not) stores its barycenter, number of points and un-centered covariance, then each time the cell is entirely included inside the ball centered at the query point p and with radius r, the cell barycenter and coefficients should be considered instead of computing the covariance from the set of neighboring points. More specifically, when computing the local regression plane, we only need to compute the neighborhood covariance Σ and the barycenter o from the set of N_r neighboring points. The covariance being a symmetric matrix, we only store its upper coefficients.

When building a tree, values must be stored in each cell. Each time a point p is added to the octree, each cell C it traverses when going down in the tree until reaching the leaf, updates part of the un-centered covariance matrix A and the unnormalized barycenter q:

$$\begin{aligned}
a_{00} &= a_{00} + p.x^2 \\
a_{01} &= a_{01} + p.x \cdot p.y \\
a_{02} &= a_{02} + p.x \cdot p.z \\
a_{11} &= a_{11} + p.y^2 \\
a_{12} &= a_{12} + p.y \cdot p.z \\
a_{22} &= a_{22} + p.z^2
\end{aligned}$$

and unnormalized barycenter coefficients

$$\begin{aligned}
q.x &= q.x + p.x \\
q.y &= q.y + p.y \\
q.z &= q.z + p.z,
\end{aligned}$$

and the number of points $N = N + 1$. When a query point p is given and we look at cells intersecting its r-ball, then the following alternative decisions can be made

- If the cell is included in the ball neighborhood, the coefficients of A (respectively q) are added to the coefficients of Σ (resp o) and the number of neighbors $N_r = N_r + N$;
- If the cell is not included in the ball neighborhood and does not intersect the ball neighborhood, discard it;

- If the cell is not included in the ball neighborhood but intersects the ball then:
  - If the cell is a leaf then look individually at the points it contains and update Σ, o, N_r accordingly;
  - If the cell is not a leaf then process all of its children cells using the same method.

At the end we have the upper coefficients of the un-centered covariance matrix Σ, the sum of all neighbor coordinates O and the number of neighbors and the final values can be deduced as

$$
\begin{aligned}
o.x &= o.x/N_r \\
o.y &= o.y/N_r \\
o.z &= o.z/N_r \\
\Sigma_{00} &= \Sigma_{00} - N_r \cdot o.x^2 \\
\Sigma_{01} &= \Sigma_{01} - N_r \cdot o.x \cdot o.y \\
\Sigma_{02} &= \Sigma_{02} - N_r \cdot o.x \cdot o.z \\
\Sigma_{11} &= \Sigma_{11} - N_r \cdot o.y^2 \\
\Sigma_{12} &= \Sigma_{12} - N_r \cdot o.y \cdot o.z \\
\Sigma_{22} &= \Sigma_{22} - N_r \cdot o.z^2
\end{aligned}
$$

Remembering that Σ is symmetric, we obtain the final centered covariance and barycenter. This simplification explains why when using a high depth value, the computation time can be reduced. Its drawback is that high depth values require a larger memory use (by increasing the octree depth by 1, 8 times more pointers are created!).

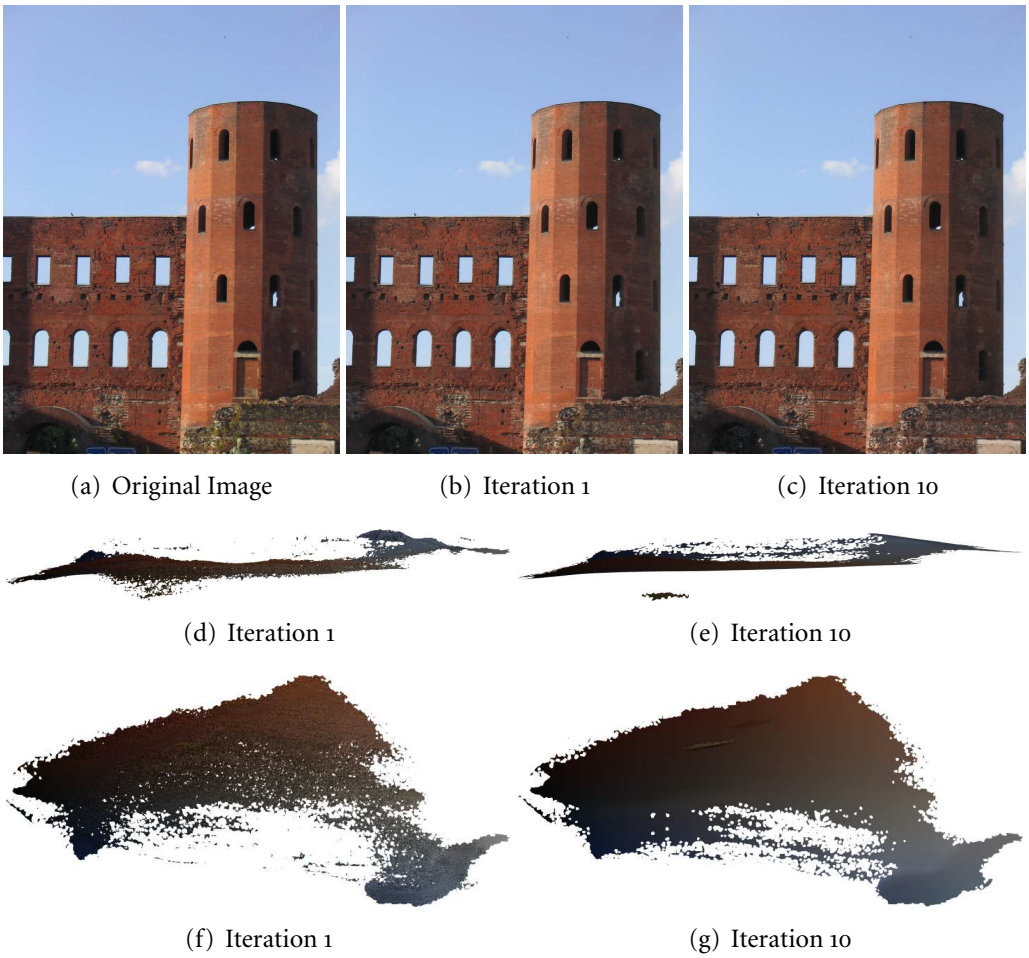
8.3 Results

Result of color cloud filtering are shown on figs 8.2, 8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 8.10, 8.12.

One can see on fig 8.2 that the perception of the image barely changes, over the iterations. By looking closely, however, one can notice that some colors disappeared (fig. 8.3).

Conclusion

This chapter verified on numerous examples that applying the regression plane projection operator to a color cloud did not change a lot the perception of the image



(a) Original Image (b) Iteration 1 (c) Iteration 10

(d) Iteration 1 (e) Iteration 10

(f) Iteration 1 (g) Iteration 10

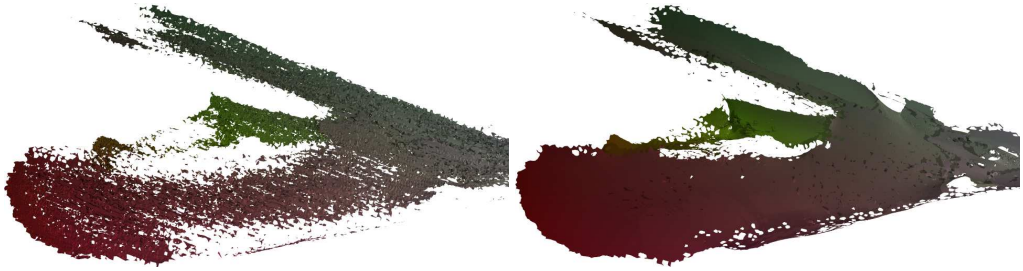
Figure 8.2: Evolution of image "Turin": original image and image evolution (top row) and two views of its color cloud (two last rows) after 1 and 10 iterations.

therefore comforting the theory of [BLM10]. The filter is very fast when applied to small size images (below 1000×1000), yet when the image size increases, a good compromise must be found between time efficiency and memory use. A useful addition to [BLM10] is that now the noisy color cloud can even be meshed at all scale by scale space meshing. Experimental evidence shows that the mesh is already visible after only one filtering iteration.

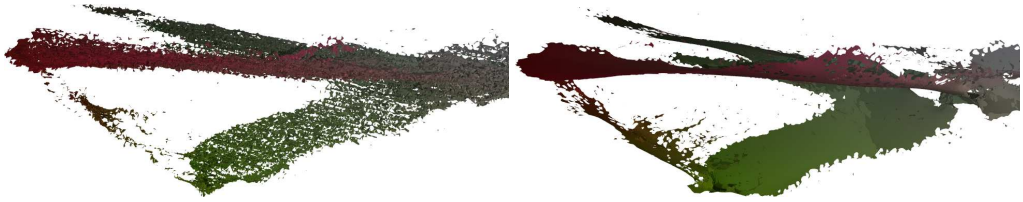
(a) Detail of the initial image (b) Detail of the filtered image

Figure 8.3: Color loss in the image filtering.

(a) Original Image (b) Iteration 1 (c) Iteration 6

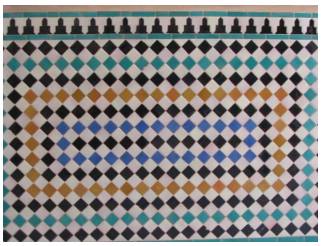


(d) Iteration 1 (e) Iteration 6

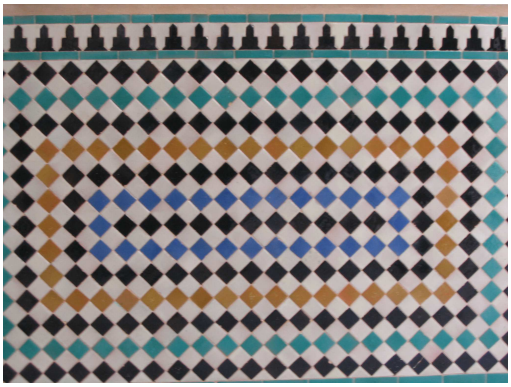


(f) Iteration 1 (g) Iteration 6

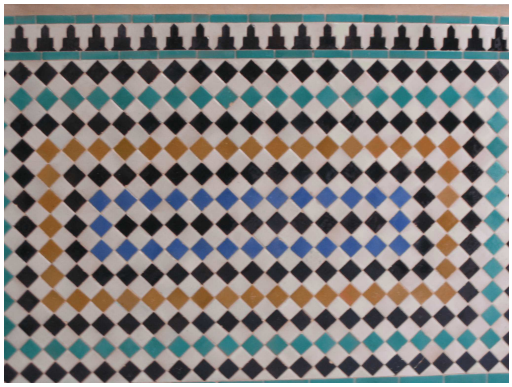
Figure 8.4: Evolution of a color image with iterative projection filter (radius 20) (top row) and color cloud (bottom row)



(a) Original Image



(b) Iteration 1



(c) Iteration 6

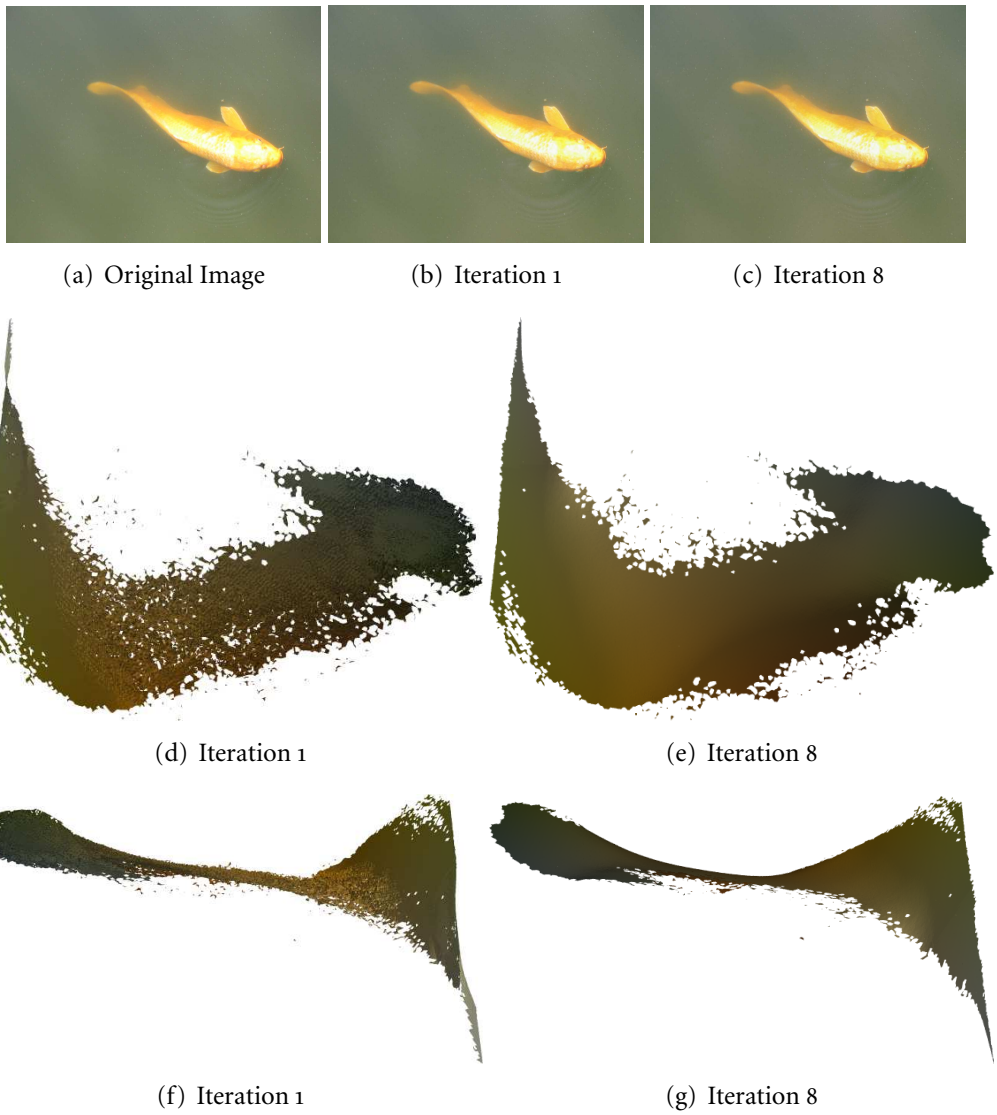
(d) Iteration 1

(e) Iteration 6

(f) Iteration 1

(g) Iteration 6

Figure 8.5: Evolution of a color image with iterative projection filter (radius 20).

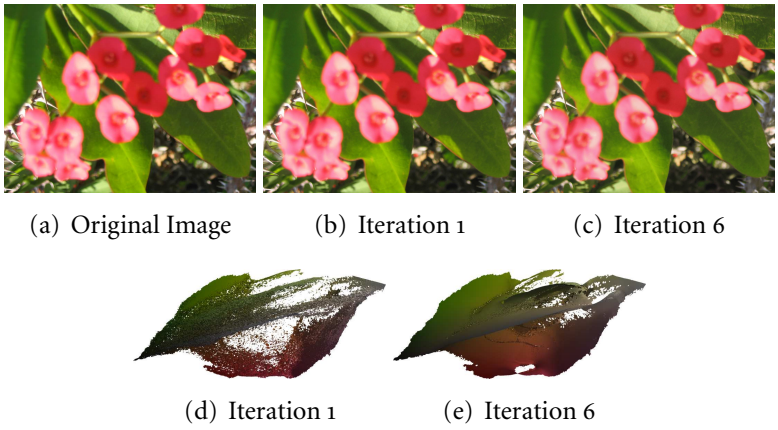


(a) Original Image (b) Iteration 1 (c) Iteration 8

(d) Iteration 1 (e) Iteration 8

(f) Iteration 1 (g) Iteration 8

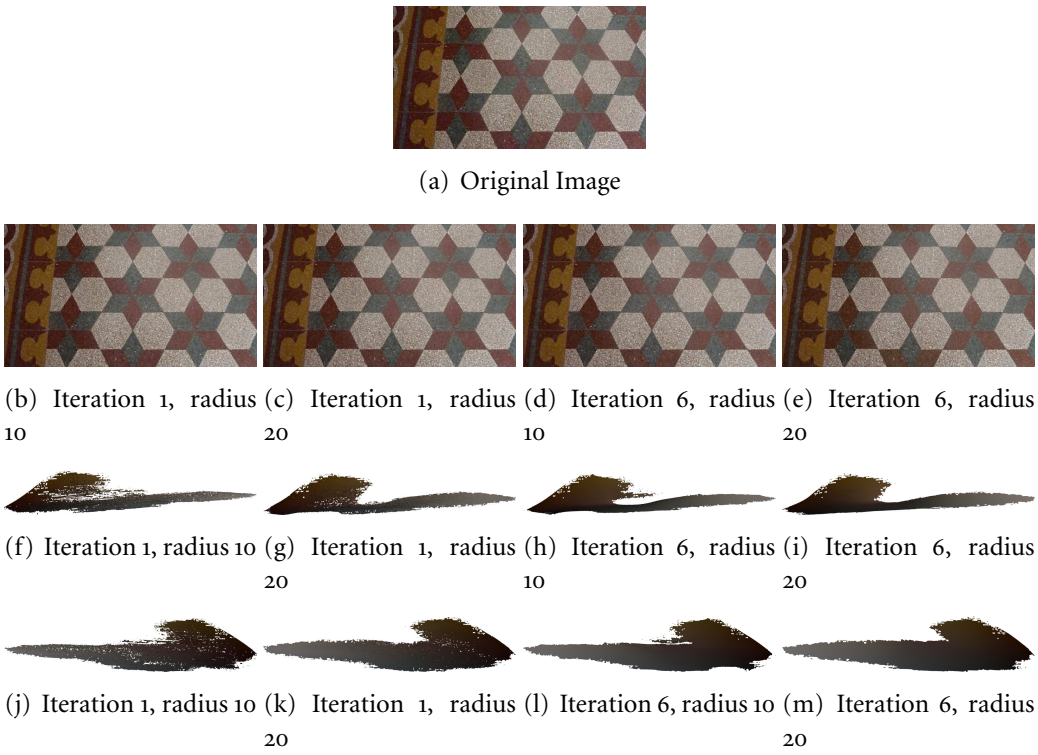
Figure 8.6: Evolution of image “fish” with iterative projection filter (radius 20).



(a) Original Image (b) Iteration 1 (c) Iteration 6

(d) Iteration 1 (e) Iteration 6

Figure 8.7: Evolution of image “flowers” with iterative projection filter (radius 20).



(a) Original Image

(b) Iteration 1, radius 10 (c) Iteration 1, radius 20 (d) Iteration 6, radius 10 (e) Iteration 6, radius 20

(f) Iteration 1, radius 10 (g) Iteration 1, radius 20 (h) Iteration 6, radius 10 (i) Iteration 6, radius 20

(j) Iteration 1, radius 10 (k) Iteration 1, radius 20 (l) Iteration 6, radius 10 (m) Iteration 6, radius 20

Figure 8.8: Comparison of the filtering between 5 iterations of the projection filter with radius 10 and 20.

(a) Original Fresco Image

(b) Iteration 0

(c) Iteration 5

(d) Iteration 0

(e) Iteration 5

(f) Iteration 0

(g) Iteration 5

Figure 8.9: Evolution of image “fresco” with iterative projection filter (radius 20).

(a) Original "Orange tree" Image

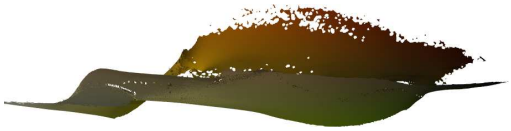
(b) Iteration 0

(c) Iteration 5

(d) Iteration 0

(e) Iteration 5

(f) Iteration 0

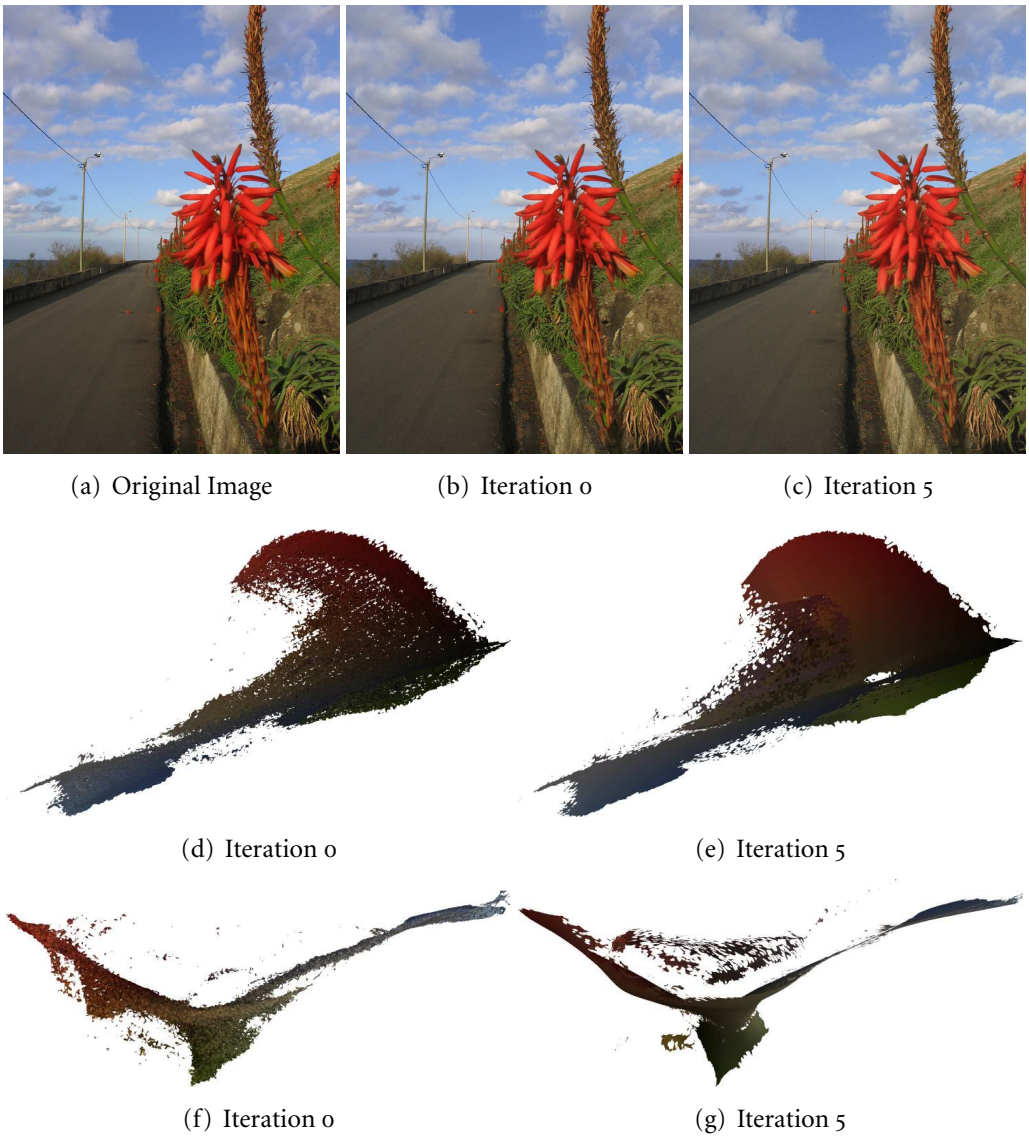


(g) Iteration 5

Figure 8.10: Evolution of image "Orange tree" with iterative projection filter (radius 20).

(a) Detail of the initial image (b) Detail of the filtered image

Figure 8.11: Color change.



(a) Original Image (b) Iteration 0 (c) Iteration 5

(d) Iteration 0 (e) Iteration 5

(f) Iteration 0 (g) Iteration 5

Figure 8.12: Evolution of image “road” with iterative projection filter (radius 20).

Chapter 9

Conclusion

In this thesis, a whole point cloud processing pipeline from the acquisition to the final mesh segmentation and extraction was studied. Yet the field of high precision shape representation is still wide open. Here we summarize some of the possible improvements and future work.

The method introduced proved to be very versatile because it was not only able to process high precision surfaces, but was also able to deal with more noisy data such as color clouds. In the continuation of this work, different types of data and improvements could be made to get a still better virtual representation of the acquired object. It seemed to us that, little by little, this work was clearing the road toward the high precision visualization and exploration of fine arts and archaeological objects acquired by a high precision laser. The field of cultural heritage digitization is already a very active (through the 3d-coform project[1] for example), but not to the precision we now get. This work opens, in some sense, the hope of a digital conservation and of an improved and more versatile display to amateurs and professionals of these sometimes fragile small pieces In particular the digital magnification seems to be an efficient visualization tool for high precision scans.

9.1 Another type of data: 3D from stereo

The existence of passive light acquisition device was already mentioned in this work. From a set of stereo-rectified images, one can extract points on the surface of the object, either by calibrating one camera and rectifying several images, or by using directly a stereo camera. An interesting goal would be to build the stereo reconstruction for the same objects that we used with the laser scanner. Then precisions of active and passive light acquisition devices could and should be compared and, hopefully, lead to technologically very needed cross-calibration tools of 3D devices.

9.2 High precision registration

Our laser scanner was somewhat limited in terms of the size of the acquired objects. In order to be still able to acquire larger objects with the same precision,

[1] http://www.3dcoform.eu/

we would need to move the object between different acquisitions. All the acquired parts should then be registered. The problem of shape registration has been studied already, yet in most cases the mesh was then built by the Poisson method, which actually hides the registration artifacts rather than it corrects them. A reasonable question is: can we register point sets well enough so that our reconstruction method can be used? Many good descriptors have been proposed to perform the registration and this could lead to test their precision.

9.3 Color reprojection

If the final goal is a high precision digital reproduction of art pieces, in other terms a digital Tanagra process, the visual information (the object's texture) will have to be fused with its 3D shape. It is therefore a stimulating problem to realize a very accurate reprojection of the color on the virtual object from a set of pictures. However, the pictures should be first registered to the 3D object, leading to another type of registration: the 3D-2D registration, which has already been investigated but never for data of this accuracy. The color information should also not be perturbed by shadows nor by geometric textures (such as those are acquired by the laser).

While these problems seem to be still wide open, considering them raises the hope of a quality virtual representation.

Appendix A

Computational Issues

Contents

Abstract: This appendix shortly explains the implementation choices of the algorithms presented in the thesis.

A.1 Classifying points

All along this work an efficient fixed-radius neighbor finding algorithm was needed. Indeed this is one of the trickiest problems in surface processing: given a point p find the coordinates of all samples lying within a given distance of p. Because of sampling irregularities it may happen that this neighborhood contains either no other point than p or the whole set.

The very naive implementation would need to traverse once the whole set of N points for each of the N points, yielding an algorithm with N^2 complexity. In case of point sets with more than a million points it yields prohibitive computation time. This is why, tree structures are usually used to partition the space into cells. In a nutshell, this partition permits to avoid traversing cells that are too far away from the query point ([Sam90]).

Here an octree was used: it simply divides the bounding box of side L into eight cells of size $L/2$. The splitting point is of course the center of the parent cell. All points are included in *leaf cells*, i.e. cells with no child that have depth d, where d is the octree depth specified by the user.

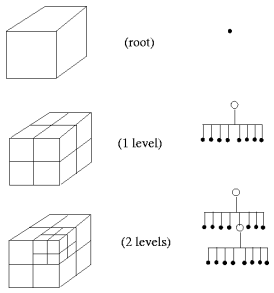


Figure A.1: An octree

Given a set of points and a depth d, the octree building proceeds as follows:

- An initial root node is built so that it contains all points;
- When a point p is added to a cell C:
  - If the cell has depth d it is a leaf and the point is simply added to the list of point of C;
  - Otherwise, look for the cell child C_i that would contain the point;

– If C_i does not exist create it and add the point to it.

This way, only nodes *actually* containing points are created.

A.2 Iterating over the octree

Iterating over the octree means from a given point position find a set of neighbors. In this work *radius fixed neighborhoods* are used meaning that given a point p and a point set $\mathcal{M}_S$, the set of p's neighbors is the subset of $\mathcal{M}_S$ such that:

$$\{m \in \mathcal{M}_S | \|p - m\|^2 < r\}$$

.

The iterating method is basically the one from [FP02]. Given a point p, a radius r and a set $\mathcal{M}_S$ it proceeds as follows:

- In case it is not given, the leaf cell C containing p is found.
- Going upwards in the tree, it finds the cell C_0 containing C and whose length L is such that $\frac{L}{4} < r \leq \frac{L}{2}$
- Then the set of neighbors is included in C_0 and C_0's neighboring cells at the same depth.
- For each of those cells:
  - Look at all non-empty children nodes
  - If the hypercube maximum distance to p is below r then all points included in its descendants are added to the set of neighbors
  - If the hypercube maximum distance to p is above r and the minimum distance is below r then, if the node is not a leaf, the node's children are explored recursively. If it is a leaf then all the point distances are computed and compared to r.
  - If the hypercube minimum distance to p is above r the cell is not explored any further.

The whole neighboring cell search is made faster using Location Codes (see [FP02] for details).

A.3 A single octree to deal with an evolving point set

Since the pointset evolves with filtering iterations, an evolving structure has to be built. This is done by simply considering that each cell contains more than 1 set of points. Since the initial pointset must be remembered, it is necessary to store 3 sets per leaf: the set of initial points, the set of points filtered at iteration N and the set of points filtered at step $N + 1$. Then when performing neighbor-search, the appropriate set should be chosen.

- When filtering set 0, the results are stored in set 1
- When filtering set 1, neighbor-search is done in set 1, the results are stored in set 2 and set 1 is emptied
- When filtering set 2, neighbor-search is done in set 2, the results are stored in set 1 and set 2 is emptied

Each point links to the point of set 0 from which it originated.

A.4 Implementation

All the algorithms presented in this thesis were implemented using C++ and the Standard Template Library (STL). It uses the Template Numerical Toolkit and JAMA-C++ libraries[1] for Principal Component Analysis and system solving.

Reading/Writing images for color cloud filtering for example was done using the CImg library [2].

Finally, reading and writing mesh in Stanford PLY format used rply [3].

For perenity reasons, files are always written in ascii PLY and not binary PLY, in order to be able to access the information on any computer in the next years, at the cost of bigger files.

Throughout this thesis renderings have been computed using the POV-RAY program[4]. No texture was used (the triangles were colored in white) and a simulated diffuse light was set.

[1] http://math.nist.gov/tnt/index.html
[2] http://cimg.sourceforge.net/
[3] http://w3.impa.br/ diego/software/rply/
[4] Persistence of Vision Raytracer http://www.povray.org

Bibliography

[AA09] Alexa M., Adamson A.: Interpolatory point set surfaces convexity and hermite data. *ACM Trans. Graph. 28* (May 2009), 20:1–20:10. 30

[AB98] Amenta N., Bern M.: Surface reconstruction by Voronoi filtering. In *SCG '98: Proceedings of the fourteenth annual symposium on Computational geometry* (USA, 1998), ACM, pp. 39–48. 23, 60

[ABCO*01] Alexa M., Behr J., Cohen-Or D., Fleishman S., Levin D., Silva C. T.: Point set surfaces. In *Proc. Vis '01* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 21–28. 121

[ABCO*03] Alexa M., Behr J., Cohen-Or D., Fleishman S., Levin D., Silva C. T.: Computing and rendering point set surfaces. *IEEE TVCG 9*, 1 (2003), 3–15. 30, 121

[ABE99] Amenta N., Bern M., Eppstein D.: Optimal point placement for mesh smoothing. *J. Algorithms 30*, 2 (1999), 302–322. 24

[ABK98] Amenta N., Bern M., Kamvysselis M.: A new voronoi-based surface reconstruction algorithm. In *SIGGRAPH '98* (USA, 1998), ACM, pp. 415–421. 23, 60

[ACK01] Amenta N., Choi S., Kolluri R.: The powercrust. In *Sixth ACM Symposium on Solid Modeling and Applications* (USA, 2001), ACM Press, pp. 249–260. 23

[ACSD*03] Alliez P., Cohen-Steiner D., Devillers O., Lévy B., Desbrun M.: Anisotropic polygonal remeshing. *ACM Trans. Graph. 22*, 3 (2003), 485–493. 24

[ACSTD07] Alliez P., Cohen-Steiner D., Tong Y., Desbrun M.: Voronoi-based variational reconstruction of unoriented point sets. In *SGP '07* (Switzerland, 2007), Eurographics, pp. 39–48. 23, 76, 117

[AK04] Amenta N., Kil Y. J.: Defining point-set surfaces. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (USA, 2004), ACM Press, pp. 264–270. 121

[AKSA09] Abbasinejad F., Kil Y. J., Sharf A., Amenta N.: Rotating scans for systematic error removal. *Computer Graphics Forum 28* (2009), 1319–1326. 60

[AMCO08] Aiger D., Mitra N. J., Cohen-Or D.: 4-points congruent sets for robust surface registration. *ACM Transactions on Graphics 27*, 3 (2008), #85, 1–10. 59

[AMD02] Alliez P., Meyer M., Desbrun M.: Interactive geometry remeshing. *ACM Trans. Graph. 21*, 3 (2002), 347–354. 24, 30

[AUGA05] Alliez P., Ucelli G., Gotsman C., Attene M.: *Recent Advances in Remeshing of Surfaces*. Part of the state-of-the-art report, AIM@SHAPE EU network of excellence, 2005. 24

[AVDI03] Alliez P., Verdière E. C. d., Devillers O., Isenburg M.: Isotropic surface remeshing. In *SMI '03: Proceedings of the Shape Modeling International 2003*. IEEE Computer Society, Washington, DC, USA, 2003, p. 49. 24

[BA83] Burt P. J., Adelson E. H.: A multiresolution spline with application to image mosaics. *ACM Trans. Graph. 2*, 4 (1983), 217–236. 61, 62

[BA05] Belyaev A., Anoshkina E.: *Mathematics of Surfaces XI, IMA Conference on the Mathematics of Surfaces 2005*. Springer, 2005, ch. Detection of Surface Creases in Range Data. 25, 96

[Bau02] Baumberg A.: Blending images for texturing 3d models. In *Proc. Conf. on British Machine Vision Association* (2002), pp. 404–413. 61

[BB97] Bernardini F., Bajaj C. L.: Sampling and reconstructing manifolds using alpha-shapes. In *In Proc. 9th Canad. Conf. Comput. Geom* (1997). 23

[BBBK09] Bronstein A. M., Bronstein M. M., Bruckstein A. M., Kimmel R.: Partial similarity of objects, or how to compare a centaur to a horse. *Int. J. Comput. Vision 84*, 2 (2009), 163–183. 22

[BBK03] Bronstein A., Bronstein M., Kimmel R.: Expression-invariant 3d face recognition. In *Proc. Audio- and Video-based Biometric Person Authentication (AVBPA)* (2003), pp. 62–69. 22

[BBK05a] Bronstein A., Bronstein M., Kimmel R.: Expression-invariant face recognition via spherical embedding. In *ICIP05* (2005), pp. III: 756–759. 22

[BBK05b] Bronstein A. M., Bronstein M. M., Kimmel R.: Isometric embedding of facial surfaces into $\mathbf{s}^3$. In *Scale Space and PDE Methods in Computer Vision*, vol. 3459 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005, pp. 622–631. 22

[BBK06] Bronstein A. M., Bronstein M. M., Kimmel R.: Efficient computation of isometry-invariant distances between surfaces. *SIAM J. Sci. Comput. 28*, 5 (2006), 1812–1836. 21

[BBK07] Bronstein A. M., Bronstein M. M., Kimmel R.: Expression-invariant representations of faces. *IEEE TRANS. PAMI* (2007), 1042–1053. 22

[BBK09] Bronstein A. M., Bronstein M. M., Kimmel R.: Topology-invariant similarity of nonrigid shapes. *Int. J. Comput. Vision 81*, 3 (2009), 281–301. 22

[BBK*10] Bronstein A. M., Bronstein M. M., Kimmel R., Mahmoudi M., Sapiro G.: A gromov-hausdorff framework with diffusion geometry for topologically-robust non-rigid shape matching. *IJCV 89*, 2-3 (2010), 266–286. 22

[BC94] Berkmann J., Caelli T.: Computation of surface geometry and segmentation using covariance techniques. *IEEE PAMI 16*, 11 (1994), 1114–1116. v, 31, 115, 119, 125, 126, 128, 144

[BCM01] Ballester C., Caselles V., Monasse P.: *The Tree of Shapes of an Image*. Tech. rep., 2001. 97, 99

[BCM05] Buades A., Coll B., Morel J. M.: A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation 4*, 2 (2005), 490–530. 26

[BCM06] Buades A., Coll B., Morel J.-M.: Neighborhood filters and pdes. *Numer. Math. 105*, 1 (2006), 1–34. 122

[BDE96] Barequet G., Dickerson M., Eppstein D.: On triangulating three-dimensional polygons. In *SCG '96: Proceedings of the twelfth annual symposium on Computational geometry* (USA, 1996), ACM, pp. 38–47. r. 77

[BK04] Botsch M., Kobbelt L.: A remeshing approach to multiresolution modeling. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (USA, 2004), ACM, pp. 185–192. 24

[BL07] Brown M., Lowe D. G.: Automatic panoramic image stitching using invariant features. *Int. J. Comput. Vision 74*, 1 (2007), 59–73. 61

[BLM10] Buades T., Lisani J., Morel J.: On the distribution of colors in natural images. preprint MAP5, February 2010. 11, 150, 151, 154

[BM92] Besl P. J., McKay N. D.: A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell. 14*, 2 (1992), 239–256. 22, 57

[BMR*99] Bernardini F., Mittleman J., Rushmeier H., Silva C., Taubin G.: The ball-pivoting algorithm for surface reconstruction. *IEEE TVCG 5* (1999), 349–359. 7, 23, 42, 60, 66

[BNK02] Borodin P., Novotni M., Klein R.: Progressive gap closing for mesh repairing. In *Advances in Modelling, Animation and Rendering*, Vince J., Earnshaw R., (Eds.). Springer Verlag, July 2002, pp. 201–213. 77

[Boi84] Boissonnat J.-D.: Geometric structures for three-dimensional shape representation. *ACM Trans. Graph. 3*, 4 (1984), 266–286. 23

[BPK*08] Botsch M., Pauly M., Kobbelt L., Alliez P., Lévy B.: Geometric modeling based on polygonal meshes. In *Eurographics Tutorial* (2008). 79

[BR04] Brown B., Rusinkiewicz S.: Non-rigid range-scan alignment using thin-plate splines. In *3DPVT'04* (2004). printed. 22, 57, 59

[BR07] Brown B., Rusinkiewicz S.: Global non-rigid alignment of 3-D scans. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26*, 3 (Aug. 2007). 22, 57, 59, 71, 72

[BRM*02] Bernardini F., Rushmeier H., Martin I., Mittleman J., Taubin G.: Building a digital model of michelangelo's florentine pieta. 59–67. printed (dossier recalage). 19

[BS95] Barequet G., Sharir M.: Filling gaps in the boundary of a polyhedron. *Comput. Aided Geom. Des. 12*, 2 (1995), 207–229. 77

[BS97] Benjemaa R., Schmitt F.: Fast global registration of 3d sampled surfaces using a multi-z-buffer technique. In *Image and Vision Computing* (1997), pp. 113–120. 59

[BSK05a] Bendels G. H., Schnabel R., Klein R.: Detail-preserving surface inpainting. In *The 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST)* (Nov. 2005), Eurographics Association, Eurographics, pp. 41–48. 79

[BSK05b] Bendels G. H., Schnabel R., Klein R.: Fragment-based surface inpainting. In *Eurographics Symposium on Geometry Processing 2005* (July 2005), Desbrun M., Pottmann H., (Eds.), Eurographics. 79

[BSK06] Bendels G. H., Schnabel R., Klein R.: Detecting holes in point set surfaces. *Journal of WSCG 14* (Feb. 2006). 80

[BSW09] Belkin M., Sun J., Wang Y.: Constructing laplace operator from point clouds in rd. In *Proc. SODA '09* (USA, 2009), SIAM, pp. 1031–1040. 21, 30, 62, 119

[BWS*09] Brunton A., Wuhrer S., Shu C., Bose P., Demaine E.: Filling holes in triangular meshes by curve unfolding. In *SMI 2009* (2009), IEEE, (Ed.), pp. 66–72. 78

[CA97] Crossno P., Angel E.: Isosurface extraction using particle systems. In *IEEE Visualization '97* (1997), Yagel R., Hagen H., (Eds.), pp. 495–498. 23

[CBC*01] Carr J. C., Beatson R. K., Cherrie J. B., Mitchell T. J., Fright W. R., McCallum B. C., Evans T. R.: Reconstruction and representation of 3d objects with radial basis functions. 67–76. 23, 77, 117

[Che95] Cheng Y.: Mean shift, mode seeking, and clustering. *IEEE PAMI 17*, 8 (1995), 790–799. 37

[CJ97] Chua C. S., Jarvis R.: Point signatures: A new representation for 3d object recognition. *International Journal of Computer Vision 25*, 1 (October 1997), 63–85. 22

[CJ03] Chalmovianský P., Jüttler B.: *Mathematics of Surfaces*. Springer, 2003, ch. Filling Holes in Point Clouds, pp. 196–212. 80

[CL96] Curless B., Levoy M.: A volumetric method for building complex models from range images. In *SIGGRAPH '96* (USA, 1996), ACM Press, pp. 303–312. 45, 59, 60, 117

[CM92] Chen Y., Medioni G.: Object modeling by registration of multiple range images. *Image Vision Comput. 10*, 3 (1992), 145–155. 57

[CMS05] Cao F., Musé P., Sur F.: Extracting meaningful curves from images. *J. Math. Imaging Vis. 22*, 2-3 (2005), 159–181. 97

[CP03] Cazals F., Pouget M.: Estimating differential quantities using polynomial fitting of osculating jets. In *SGP '03* (Switzerland, 2003), Eurographics, pp. 177–187. 38, 96, 121

[CPZ07] Che W., Paul J.-C., Zhang X.: Lines of curvature and umbilical points for implicit surfaces. *Comput. Aided Geom. Des. 24*, 7 (2007), 395–409. 97

[CR03] Chui H., Rangarajan A.: A new point matching algorithm for non-rigid registration. *Comput. Vis. Image Underst. 89* (2003), 114–141. 22, 60

[CSA00] Carr H., Snoeyink J., Axen U.: Computing contour trees in all dimensions. In *SODA '00* (Philadelphia, PA, USA, 2000), SIAM, pp. 918–926. 97

[CSAD04] Cohen-Steiner D., Alliez P., Desbrun M.: Variational shape approximation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (USA, 2004), ACM Press, pp. 905–914. 24

[CSM03] Cohen-Steiner D., Morvan J.-M.: Restricted delaunay triangulations and normal cycle. In *Proc. SCG '03* (USA, 2003), ACM, pp. 312–321. 31, 118

[DB06] Donoser M., Bischof H.: Efficient maximally stable extremal region (mser) tracking. In *IEEE CVPR 2006* (2006), vol. 1, pp. 553–560. 97, 99

[DDSS10] Digne J., Dimiccoli M., Sabater N., Salambier P.: *The Handbook of Mathematical Methods in Imaging*. Springer Verlag, to appear in 2010, ch. Neighborhood filters and the recovery of 3D information. 12

[DFRS03] DeCarlo D., Finkelstein A., Rusinkiewicz S., Santella A.: Suggestive contours for conveying shape. *ACM Trans. Graph. 22*, 3 (2003), 848–855. 97

[DIHOS07] Daniels II J., Ha L., Ochotta T., Silva C.: Robust smooth feature extraction from point clouds. In *SMI '07* (Washington, DC, USA, 2007), IEEE, pp. 123–136. 96

[DIOHS08] Daniels II J., Ochotta T., Ha L. K., Silva C. T.: Spline-based feature curves from point-sampled geometry. *Vis. Comput. 24*, 6 (2008), 449–462. 25, 96

[DM10] Digne J., Morel J.-M.: A numerical analysis of raw point cloud smoothing. submitted. 11, 12

[DMAL10] Digne J., Morel J.-M., Audfray N., Lartigue C.: High fidelity scan merging. *CGF 29*, 5 (2010), 1643–1651. SGP2010. 8, 12, 148

[DMAMS10] Digne J., Morel J.-M., Audfray N., Mehdi-Souzani C.: The level set tree on meshes. In *Proceedings of the Fifth International Symposium on. 3D Data Processing, Visualization and Transmission* (Paris, France, May 2010). 9, 12

[DMGL01] Davis J., Marschner S. R., Garr M., Levoy M.: Filling holes in complex surfaces using volumetric diffusion. In *In First International Symposium on 3D Data Processing, Visualization, and Transmission* (2001), pp. 428–438. 76

[DMM01] Desolneux A., Moisan L., Morel J.-M.: Edge detection by helmholtz principle. *J. Math. Imaging Vis. 14*, 3 (2001), 271–284. 97, 109

[DMMSL09] Digne J., Morel J.-M., Mehdi-Souzani C., Lartigue C.: Scale space meshing of raw data point sets. preprint CMLA 2009-30 - ENS Cachan, October 2009. 7, 12, 62, 106, 108, 134, 135, 145, 148

[DMSB99] Desbrun M., Meyer M., Schroder P., Barr A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99* (USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 317–324. 118

[DSGV01] Dias P., Sequeira V., Goncalves J., Vaz F.: Fusion of intensity and range data for improved 3d models. pp. III: 1107–1110. 18

[DVVR06] Demarsin K., Vanderstraeten D., Volodine T., Roose D.: Detection of closed sharp feature lines in point clouds for reverse engineering applications. In *GMP06* (2006), pp. 571–577. 96

[Ede93] Edelsbrunner H.: The union of balls and its dual shape. 218–231. 23

[EM94] Edelsbrunner H., Mücke E. P.: Three-dimensional alpha shapes. *ACM Trans. Graph. 13*, 1 (1994), 43–72. 23

[FCOS05] Fleishman S., Cohen-Or D., Silva C. T.: Robust moving least-squares fitting with sharp features. *ACM Trans. Graph. 24*, 3 (2005), 544–552. 30, 121

[FDCO03] Fleishman S., Drori I., Cohen-Or D.: Bilateral mesh denoising. *ACM Trans. Graph. 22*, 3 (2003), 950–953. 70

[FHK*04] Frome A., Huber D., Kolluri R., BÃ$\frac{1}{4}$low T., Malik J.: Recognizing objects in range data using regional point descriptors. In *EUROPEAN CONFERENCE ON COMPUTER VISION* (2004), pp. 224–237. 22

[FP02] Frisken S. F., Perry R.: Simple and efficient traversal methods for quadtrees and octrees. *Journal of graphics tools 7(3)*, 3 (2002), 1–11. 167

[GG07] Guennebaud G., Gross M.: Algebraic point set surfaces. *ACM Trans. Graph. 26* (2007). 30, 45

[GH00] Gross M., Hubeli A.: *Eigenmeshes*. Tech. rep., ETH Zurich, 2000. 118

[GIRL03] Gelfand N., Ikemoto L., Rusinkiewicz S., Levoy M.: Geometrically stable sampling for the icp algorithm. In *Proc. 3DIM 2003* (2003), pp. 260–267. 59

[GKS00] Gopi M., Krishnan S., Silva C.: Surface reconstruction based on lower dimensional localized delaunay triangulation. *CGF 19*, 3 (September 2000), 467–478. 30

[GLWYT06] Guo T.-Q., Li J.-J., Weng J.-G., Y.-T. Z.: Filling holes in complex surfaces using oriented voxel diffusion. In *Int. Conf. on Machine Learing and Cybernetics* (August 2006), IEEE, (Ed.), pp. 4370–4375. 77

[GTE*06] Gois J. P., Tejada E., Etiene T., Nonato L. G., Castelo A., Ertl T.: Curvature-driven modeling and rendering of point-based surfaces. *Computer Graphics and Image Processing, Brazilian Symposium on* (2006), 27–36. 30

[GWM01] Gumhold S., Wang X., McLeod R.: Feature extraction from point clouds. In *Proc. 10th International Meshing Roundtable* (2001). 25, 96

[HDD*92] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: Surface reconstruction from unorganized points. In *SIGGRAPH '92* (USA, 1992), ACM Press, pp. 71–78. 23, 29, 42, 76, 117

[HDD*93] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: Mesh optimization. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (USA, 1993), ACM, pp. 19–26. 24

[HG01] Hubeli A., Gross M.: Multiresolution feature extraction for unstructured meshes. In *VIS '01: Proceedings of the conference on Visualization '01* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 287–294. 121

[HID95] Hebert M., Ikeuchi K., Delingette H.: A spherical representation for recognition of free-form surfaces, 1995. 22

[HMG00] Hubeli A., Meyer K., Gross M.: Mesh edge detection. In *Proc. Workshop Lake Tahoe* (Lake Tahoe City, California, USA, January 2000). 25, 96

[Hor87] Horn B. K. P.: Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A 4*, 4 (1987), 629–642. 22, 57

[HP04] Hildebrandt K., Polthier K.: Anisotropic filtering of non-linear surface features. *CGF 23* (2004), 391–400. 31

[HPW05] Hildebrandt K., Polthier K., Wardetzky M.: Smooth feature lines on surface meshes. In *SGP '05* (Switzerland, 2005), Eurographics Association, p. 85. 96

[IFP95] Interrante V., Fuchs H., Pizer S.: Enhancing transparent skin surfaces with ridge and valley lines. In *VIS '95: Proc. 6th conf. on Visualization '95* (Washington, DC, USA, 1995), IEEE, p. 52. 25, 96

[Jas97] Jasiobedzki P.: Fusing and guiding range measurements with colour video images. In *3DIM97* (1997), pp. 13 – Applications. 18

[JDA07] Judd T., Durand F., Adelson E. H.: Apparent ridges for line drawing. *ACM Trans. Graph. 26*, 3 (2007), 19. 97

[JH97] Johnson A. E., Hebert M.: Surface registration by matching oriented points. In *In Proc. Int. Conf. on Recent Advances in 3-D Digital Imaging and Modeling* (1997), IEEE, pp. 121–128. 22

[JH99] Johnson A. E., Hebert M.: Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE PAMI 21* (1999), 433–449. 22, 59

[Joh97] Johnson A.: *Spin-images: A representation for 3-d surface matching*. Tech. rep., 1997. 22

[Ju04] Ju T.: Robust repair of polygonal models. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (USA, 2004), ACM, pp. 888–895. 77

[Jun05] Jun Y.: A piecewise hole filling algorithm in reverse engineering. *Computer-Aided Design 37*, 2 (2005), 263 – 270. 78

[Kaz05] Kazhdan M.: Reconstruction of solid models from oriented point sets. In *SGP '05* (Switzerland, 2005), Eurographics Association, p. 73. 23, 57, 60, 76, 117

[KBH06] Kazhdan M., Bolitho M., Hoppe H.: Poisson surface reconstruction. In *SGP '06* (Switzerland, 2006), Eurographics, pp. 61–70. 23, 29, 42, 48, 57, 60, 71, 72, 76, 117

[KBSS01] Kobbelt L. P., Botsch M., Schwanecke U., Seidel H.-P.: Feature sensitive surface extraction from volume data. In *SIGGRAPH '01* (USA, 2001), ACM, pp. 57–66. 23, 117

[KCVS98] Kobbelt L., Campagna S., Vorsatz J., Seidel H.-P.: Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (USA, 1998), ACM, pp. 105–114. 83

[KFR03] Kazhdan M., Funkhouser T., Rusinkiewicz S.: Rotation invariant spherical harmonic representation of 3d shape descriptors. In *SGP '03* (Switzerland, 2003), Eurographics Association, pp. 156–164. 22, 59

[KG00] Karni Z., Gotsman C.: Spectral compression of mesh geometry. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 279–286. 121

[KMA06] Kil Y., Mederos B., Amenta N.: Laser scanner super-resolution. Point Based Graphics. 60

[KNSS09] Kalogerakis E., Nowrouzezahrain D., Simari P., Singh K.: Extracting lines of curvature from noisy point cloud. *Computer-Aided Design 41*, 4 (2009), 282 – 292. Point-based Computational Techniques. 97

[KSNS07] Kalogerakis E., Simari P., Nowrouzezahrai D., Singh K.: Robust statistical estimation of curvature on discretized surfaces. In *SGP '07* (Switzerland, 2007), Eurographics, pp. 13–22. 119

[KST09] Kolomenkin M., Shimshoni I., Tal A.: On edge detection on surfaces. In *CVPR* (2009), pp. 2767–2774. 61, 98, 112

[KTN*06] Koller D., Trimble J., Najbjerg T., Gelfand N., Levoy M.: Fragments of the city: Stanford's digital forma urbis romae project. In *Proc. Third Williams Symposium on Classical Architecture, Journal of Roman Archaeology Suppl. 61* (2006), pp. pp. 237–252. 19, 108

[KZBB09] Kimmel R., Zhang C., Bronstein A. M., Bronstein M. M.: Are mser features really interesting? *IEEE PAMI* (2009). 99

[LC87] Lorensen W. E., Cline H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87* (USA, 1987), ACM Press, pp. 163–169. 23, 29, 42, 60, 77, 117

[LCOL07] Lipman Y., Cohen-Or D., Levin D.: Data-dependent mls for faithful surface approximation. In *SGP '07* (Switzerland, 2007), Eurographics, pp. 59–67. 30, 121

[LCOLTE07] Lipman Y., Cohen-Or D., Levin D., Tal-Ezer H.: Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph. 26* (2007). 24

[Lev98] Levin D.: The approximation power of moving least-squares. *Math. Comput. 67*, 224 (1998), 1517–1531. 121

[Lev03] Levin D.: Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization* (2003), Brunnett H., Mueller, (Eds.), Springer-Verlag, pp. 37–49. 21, 30, 121

[LFM96] Lengagne R., Fua P., Monga O.: Using crest lines to guide surface reconstruction from stereo. In *ICPR '96: Volume I* (USA, 1996), IEEE, p. 9. 25, 31, 96

[LGS06] Levet F., Granier X., Schlick C.: Fast sampling of implicit surfaces by particle systems. In *SMI 2006 - short paper* (jun 2006), Fast sampling of implicit surfaces by particle systems. 23

[Lie03] Liepa P.: Filling holes in meshes. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (Switzerland, 2003), Eurographics Association, pp. 200–205. 8, 78, 80, 82, 83, 84

[LMR07] Linsen L., Müller K., Rosenthal P.: Splat-based ray tracing of point clouds. In *Journal of WSCG (Proceedings on Fifteenth International Conference in Central Europe on Computer Vision - WSCG 2007)* (Plzen, Czech Republic, 2007), UNION Agency - Science Press. 21

[LMW10] Li Z., Meek D., Walton D.: Polynomial blending in a mesh hole-filling application. *Computer-Aided Design 42*, 4 (2010), 340 – 349. 79

[LP03] Linsen L., Prautzsch H.: Fan clouds: an alternative to meshes. In *Proceedings of the 11th international conference on Theoretical foundations of computer vision* (Berlin, Heidelberg, 2003), Springer-Verlag, pp. 39–57. 21

[LP05] Lange C., Polthier K.: Anisotropic smoothing of point sets. *CAGD 22*, 7 (2005), 680–692. 31, 96

[LPC*00] Levoy M., Pulli K., Curless B., Rusinkiewicz S., Koller D., Pereira L., Ginzton M., Anderson S., Davis J., Ginsberg J., Shade J., Fulk D.: The digital michelangelo project: 3d scanning of large statues. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 131–144. 18, 19

[LS81] Lancaster P., Salkauskas K.: Surfaces generated by moving least squares methods. *Mathematics of Computation 37*, 155 (1981), 141–158. 120

[LT90] Liang P., Todhunter J. S.: Representation and recognition of surface shapes in range images: a differential geometry approach. *Comput. Vision Graph. Image Process. 52*, 1 (1990), 78–109. 119, 130

[LYZ08] Li G., Ye X.-Z., Zhang S.-Y.: An algorithm for filling complex holes in reverse engineering. *Eng. with Comput. 24*, 2 (2008), 119–125. 78

[Mal07] Malassiotis S.: Snapshots: A novel local surface descriptor and matching algorithm for robust 3d surface alignment. *IEEE PAMI 29*, 7 (2007), 1285–1290. Fellow-Strintzis, Michael G. 22

[MCUP02] Matas J., Chum O., Urban M., Pajdla T.: Robust wide baseline stereo from maximally stable extremal regions. In *In British Machine Vision Conference* (2002), vol. 1, pp. 384–393. iv, 95, 97, 98, 108

[MDGD*10] Mullen P., De Goes F., Desbrun M., Cohen Steiner D., Alliez P.: Signing the Unsigned: Robust Surface Reconstruction from Raw Pointsets. In *Computer Graphics Forum: Proc. SGP 2010* (2010), vol. 29, pp. 1733–1741. 23

[MDSB02] Meyer M., Desbrun M., Schröder P., Barr A.: Discrete differential geometry operators for triangulated 2-manifolds. In *International Workshop on Visualization and Mathematics* (2002). 30, 31, 96, 118

[MG98] Monasse P., Guichard F.: Fast computation of a contrast-invariant image representation. *IEEE Trans. on Image Proc 9* (1998), 860–872. 97, 99, 102

[MK98] Ma W., Kruth J. P.: Nurbs curve and surface fitting for reverse engineering. *The International Journal of Advanced Manufacturing Technology 14* (Dec. 1998), 918–927. 21

[MMS*07] Moenning C., Mémoli F., Sapiro G., Dyn N., Dodgson N. A.: Meshless geometric subdivision. *Graph. Models 69*, 3-4 (2007), 160–179. 21

[MOG09] Mérigot Q., Ovsjanikov M., Guibas L. J.: Robust Voronoi-based Curvature and Feature Estimation. In *SIAM/ACM Joint Conference on Geometric and Physical Modeling* (San Francisco, USA, 2009). 25, 119

[MS02] Memoli F., Sapiro G.: *Distance Functions and Geodesics on Point Clouds*. Tech. rep., Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, 2002. 21

[MS04] Mémoli F., Sapiro G.: Comparing point clouds. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (USA, 2004), ACM, pp. 32–40. 21

[MS05a] Mémoli F., Sapiro G.: Distance functions and geodesics on submanifolds of $\setminus^d$ and point clouds. *SIAM Journal on Applied Mathematics 65*, 4 (2005), 1227–1260. 21

[MS05b] Mémoli F., Sapiro G.: A theoretical and computational framework for isometry invariant recognition of point cloud data. *Found. Comput. Math. 5*, 3 (2005), 313–347. 21

[MS09] Mahmoudi M., Sapiro G.: Three-dimensional point cloud recognition via distributions of geometric distances. *Graph. Models 71*, 1 (2009), 22–31. 21

[MSDA*10] Mehdi-Souzani C., Digne J., Audfray N., Lartigue C., Morel J.-M.: Feature extraction from high-density point clouds: toward automation of an intelligent 3d contact less digitizing strategy. In

Proceedings of the Computer-Aided Design and Applications 2010 conference (2010), Design C.-A., Applications, (Eds.). 12

[MSR07] Magid E., Soldea O., Rivlin E.: A comparison of gaussian and mean curvature estimation methods on triangular meshes of range image data. *CVIU 107*, 3 (2007), 139–159. 31, 117

[MTSM10] Mehra R., Tripathi P., Sheffer A., Mitra N. J.: Visibility of noisy point cloud data. *Comput. Graph. 34*, 3 (2010), 219–230. 21

[MWP96] Maekawa T., Wolter F.-E., Patrikalakis N. M.: Umbilics and lines of curvature for shape interrogation. *Comput. Aided Geom. Des. 13*, 2 (1996), 133–161. 97

[MYF99] Mostafa M., Yamany S., Farag A.: Integrating shape from shading and range data using neural networks. In *CVPR99* (1999), pp. II: 15–20. 18

[MZFC09] Meinhardt E., Zacur E., Frangi A., Caselles V.: 3d edge detection by selection of level surface patches. *J. Math. Imaging Vis. 34*, 1 (2009), 1–16. 97

[NC04] Najman L., Couprie M.: Quasi-linear algorithm for the component tree. In *In SPIE Vision Geometry XII* (2004), pp. 98–107. 99, 102

[NRDR05] Nehab D., Rusinkiewicz S., Davis J., Ramamoorthi R.: Efficiently combining positions and normals for precise 3d geometry. *ACM Trans. Graph. 24*, 3 (2005), 536–543. 18, 60

[NT03] Nooruddin F. S., Turk G.: Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics 9*, 2 (2003), 191–205. 77

[OBA05] Ohtake Y., Belyaev A., Alexa M.: Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, 2005), Eurographics Association, p. 149. 23

[OBS04] Ohtake Y., Belyaev B., Seidel H.-P.: Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph. 23*, 3 (2004), 609–612. 25, 96

[OGG09] Oztireli A. C., Guennebaud G., Gross M.: Feature preserving point set surfaces based on non-linear kernel regression. *CGF 28* (2009), 493–501(9). 30, 45, 121

[OW04] Omer I., Werman M.: Color lines: Image specific color representation. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on 2* (2004), 946–953. 150

[PC03] Peyre G., Cohen L.: Geodesic re-meshing and parameterization using front propagation. In *LevelSet03* (2003), pp. xx–yy. 24

[PC05] Peyre G., Cohen L.: Geodesic computation for adaptive remeshing. In *CVPR05* (2005), p. II: 1193. 24

[PC06] Peyre G., Cohen L.: Geodesic remeshing using front propagation. *IJCV 69*, 1 (August 2006), 145–156. 24

[Pet98] Peters J.: Constructing c^1 surfaces of arbitrary topology using biquadratic and bicubic splines, 1998. 21

[PGK02] Pauly M., Gross M., Kobbelt L. P.: Efficient simplification of point-sampled surfaces. In *Proc. VIS '02* (USA, 2002), IEEE, pp. 163–170. 24, 31, 119, 120, 121, 131, 144

[PKG03] Pauly M., Keiser R., Gross M.: Multi-scale feature extraction on point-sampled surfaces. In *CGF* (september 2003), vol. 22, pp. 281–289. 25, 96

[PKG06] Pauly M., Kobbelt L. P., Gross M.: Point-based multiscale surface representation. *ACM Trans. Graph. 25*, 2 (2006), 177–193. 30, 37, 38, 61, 62, 121

[PMG*05] Pauly M., Mitra N. J., Giesen J., Gross M., Guibas L. J.: Example-based 3d scan completion. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, 2005), Eurographics Association, p. 23. 79

[PMV06] Pernot J.-P., Moraru G., Véron P.: Filling holes in meshes using a mechanical model to simulate the curvature variation minimization. *Comput. Graph. 30*, 6 (2006), 892–902. 78

[PR05] Podolak J., Rusinkiewicz S.: Atomic volumes for mesh completion. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, 2005), Eurographics Association, p. 33. 77

[PS96] Pfeifle R., Seidel H.-P.: Triangular b-splines for blending and filling of polygonal holes. In *GI '96: Proceedings of the conference on Graphics interface '96* (Toronto, Ont., Canada, Canada, 1996), Canadian Information Processing Society, pp. 186–193. 78

[PSM*02] Pérez E., Salamanca S., Merchán P., Adán A., Cerrada C.: Filling holes in 3d meshes using image restoration algorithms. In *3DPVT'02* (2002). 78

[PT95] Piegl L., Tiller W.: *The NURBS book*. Springer-Verlag, London, UK, 1995. 21

[PWHY09] Pottmann H., Wallner J., Huang Q.-X., Yang Y.-L.: Integral invariants for robust geometry processing. *CAGD 26*, 1 (2009), 37–60. 31, 118

[PWY*07] Pottmann H., Wallner J., Yang Y.-L., Lai Y.-K., Hu S.-M.: Principal curvatures from the integral invariant viewpoint. *CAGD 24*, 8-9 (2007), 428–442. 31, 118

[RL01] Rusinkiewicz S., Levoy M.: Efficient variants of the icp algorithm. In *Proc. 3DIM 2001* (2001), pp. 145–152. 22, 57, 59

[Rus04] Rusinkiewicz S.: Estimating curvatures and their derivatives on triangle meshes. In *3DPVT '04* (USA, 2004), IEEE, pp. 486–493. 31, 96, 117

[SA01] Sun Y., Abidi M. A.: Surface matching by 3d point's fingerprint. In *ICCV 2001. Proceedings.* (2001), vol. 2, pp. 263–269 vol.2. 22, 59

[SAAY06] Samozino M., Alexa M., Alliez P., Yvinec M.: Reconstruction with Voronoi centered radial basis functions. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, 2006), Eurographics Association, pp. 51–60. 23

[SACO04] Sharf A., Alexa M., Cohen-Or D.: Context-based surface completion. *ACM Trans. Graph. 23*, 3 (2004), 878–887. 79

[Sam90] Samet H.: *The design and analysis of spatial data structures*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990. 166

[SCOT03] Sorkine O., Cohen-Or D., Toledo S.: High-pass quantization for mesh encoding. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (Aire-la-Ville, Switzerland, 2003), SGP '03, Eurographics Association, pp. 42–51. 61

[Sei91] Seidel R.: A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl 1* (1991), 51–64. 81

[SF04] Stylianou G., Farin G.: Crest lines for surface segmentation and flattening. *IEEE TVCG 10*, 5 (2004), 536–544. 25, 96

[SLI98] Shimada K., Liao J.-H., Itoh T.: Quadrilateral meshing with directionality control through the packing of square cells. In *Proceedings of 7th International Meshing Roundtable* (1998), pp. 61–75. 24

[SMS*03] Surazhsky T., Magid E., Soldea O., Elber G., Rivlin E.: A comparison of gaussian and mean curvatures estimation methods on triangular meshes. In *Proc. ICRA 2003* (2003). 31

[SOG09] Sun J., Ovsjanikov M., Guibas L.: A concise and provably informative multi-scale signature based on heat diffusion. 1383–1392. 22

[SOS04] Shen C., O'Brien J. F., Shewchuk J. R.: Interpolating and approximating implicit surfaces from polygon soup. In *SIGGRAPH'04* (USA, 2004), ACM Press., pp. 896–904. 30

[Tan05] Tang X.: A sampling framework for accurate curvature estimation in discrete surfaces. *IEEE TVCG 11*, 5 (2005), 573–583. 31, 119

[Tau95a] Taubin G.: Estimating the tensor of curvature of a surface from a polyhedral approximation. In *ICCV '95* (USA, 1995), IEEE, p. 902. 31, 96, 117

[Tau95b] Taubin G.: A signal processing approach to fair surface design. In *SIGGRAPH '95* (USA, 1995), ACM Press, pp. 351–358. 62, 118, 121

[TL94] Turk G., Levoy M.: Zippered polygon meshes from range images. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (USA, 1994), ACM, pp. 311–318. 77

[TRZS04] Theisel H., Rossl C., Zayer R., Seidel H.-P.: Normal based estimation of the curvature tensor for triangular meshes. In *Proc. PG' 04* (USA, 2004), IEEE, pp. 288–297. 31, 118

[TT05] Tong W.-S., Tang C.-K.: Robust estimation of adaptive tensors of curvature by tensor voting. *IEEE PAMI 27*, 3 (2005), 434–449. 118

[UH08] Unnikrishnan R., Hebert M.: Multi-scale interest regions from unorganized point clouds. In *Workshop on Search in 3D (S3D), IEEE CVPR* (2008). 30, 36, 118, 125

[VCBS03] Verdera J., Caselles V., Bertalmio M., Sapiro G.: Inpainting surface holes. In *In Int. Conference on Image Processing* (2003), pp. 903–906. 77, 80

[VSR01] Vranić D. V., Saupe D., Richter J.: Tools for 3d-object retrieval: Karhunen-loeve transform and spherical harmonics. In *Multimedia Signal Processing, 2001 IEEE Fourth Workshop on* (2001), pp. 293–298. 59

[WH94] Witkin A. P., Heckbert P. S.: Using particles to sample and control implicit surfaces. In *SIGGRAPH '94* (USA, 1994), ACM Press, pp. 269–277. 23

[Wit83] Witkin A. P.: Scale-space filtering. In *8th Int. Joint Conf. Artificial Intelligence* (1983), vol. 2, pp. 1019–1022. 29

[WO07] Wang J., Oliveira M.: Filling holes on locally smooth surfaces reconstructed from point clouds. *IVC 25*, 1 (January 2007), 103–113. 79

[WZZY08] Wang R.-F., Zhang S.-Y., Zhang Y., Ye X.-Z.: Similarity-based denoising of point-sampled surfaces. *Journal of Zhejiang University 9*, 6 (June 2008), 807–815. 26

[YBS05] Yoshizawa S., Belyaev A., Seidel H.-P.: Fast and robust detection of crest lines on meshes. In *SPM '05* (USA, 2005), ACM Press, pp. 227–232. 25, 96

[YBS06] Yoshizawa S., Belyaev A., Seidel H.-P.: Smoothing by example: Mesh denoising by averaging with similarity-based weights. In *SMI '06* (Washington, DC, USA, 2006), IEEE, p. 9. 26

[YF99] Yamany S. M., Farag A. A.: Free-form surface registration using surface signatures. In *ICCV '99* (Washington, DC, USA, 1999), IEEE, p. 1098. 22, 59

[YLHP06] Yang Y.-L., Lai Y.-K., Hu S.-M., Pottmann H.: Robust principal curvatures on multiple scales. In *SGP '06* (Aire-la-Ville, Switzerland, 2006), Eurographics, pp. 223–226. 118

[YQ07] Yang P., Qian X.: Direct computing of surface curvatures for point-set surfaces. In *Proc. PBG 07* (2007). 31, 121

[ZGL07] Zhao W., Gao S., Lin H.: A robust hole-filling algorithm for triangular mesh. *Vis. Comput. 23*, 12 (2007), 987–997. 77

[ZH99] Zhang D., Hebert M.: Harmonic maps and their applications in surface matching. In *IEEE (CVPR '99)* (1999), vol. 2. 22, 59

[ZTS09] Zatzarinni R., Tal A., Shamir A.: Relief analysis and extraction. *ACM Trans. Graph. 28*, 5 (2009), 1–9. 61, 98, 110, 112

[ZZW*08] Zeng W., Zeng Y., Wang Y., Yin X., Gu X., Samaras D.: 3d non-rigid surface matching and registration based on holomorphic differentials. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 1–14. 22

Inverse Geometry: From the raw point cloud to the 3D surface Theory and Algorithms

Abstract:

Many laser devices acquire directly 3D objects and reconstruct their surface. Nevertheless, the final reconstructed surface is usually smoothed out as a result of the scanner internal de-noising process and the offsets between different scans.

This thesis, working on results from high precision scans, adopts the somewhat extreme conservative position, not to loose or alter any raw sample throughout the whole processing pipeline, and to attempt to visualize them. Indeed, it is the only way to discover all surface imperfections (holes, offsets). Furthermore, since high precision data can capture the slightest surface variation, any smoothing and any sub-sampling can incur in the loss of textural detail.

The thesis attempts to prove that one can triangulate the raw point cloud with almost no sample loss. It solves the exact visualization problem on large data sets of up to 35 million points made of 300 different scan sweeps and more. Two major problems are addressed. The first one is the orientation of the complete raw point set, an the building of a high precision mesh. The second one is the correction of the tiny scan misalignments which can cause strong high frequency aliasing and hamper completely a direct visualization.

The second development of the thesis is a general low-high frequency decomposition algorithm for any point cloud. Thus classic image analysis tools, the level set tree and the MSER representations, are extended to meshes, yielding an intrinsic mesh segmentation method.

The underlying mathematical development focuses on an analysis of a half dozen discrete differential operators acting on raw point clouds which have been proposed in the literature. By considering the asymptotic behavior of these operators on a smooth surface, a classification by their underlying curvature operators is obtained.

This analysis leads to the development of a discrete operator consistent with the mean curvature motion (the intrinsic heat equation) defining a remarkably simple and robust numerical *scale space*. By this scale space all of the above mentioned problems (point set orientation, raw point set triangulation, scan merging, segmentation), usually addressed by separated techniques, are solved in a unified framework.

Keywords: 3D surface reconstruction, high precision point clouds, point cloud scale space