Voronoi Centred Radial Basis Functions

Marie Samozino

UNIVERSITÉ DE NICE - SOPHIA ANTIPOLIS

UFR SCIENCES

École doctorale « Sciences et Technologies de l'Information et de la Communication » de Nice - Sophia Antipolis

THÈSE

pour obtenir le titre de

Docteur en Sciences

Discipline: Automatique, Traitement du Signal et des Images

présentée par

Marie Samozino

Voronoi Centered Radial Basis Functions

Thèse dirigée par Pierre Alliez et Mariette Yvinec

soutenue le 11 juillet 2007 devant le jury composé de

Christophe Schlick	professeur	Président
Marc Alexa	professor	Rapporteur
Alexander G. Belyaev	senior scientist	Rapporteur
Raphaelle Chaine	Maitre de conférence	Examinateur
Nicolas Tsingos	Chargé de Recherche INRIA	Examinateur
Mariette Yvinec	Chargé de Recherche INRIA HDR	Directeur de thèse
Pierre Alliez	Chargé de Recherche INRIA	Directeur de thèse

Voronoi Centered Radial Basis Functions

Marie Samozino

November 3, 2008

A mes parents,
A mon frère et ma soeur

"Approche la lune du mieux que tu peux même si tu ne l’atteins pas tu seras plus proche des étoiles "

Contents

List of Figures

Abstract

This thesis considers the problem of reconstructing a surface from scattered points sampled on a physical shape. Our contribution is the development of a surface reconstruction method based on the Radial Basis Functions (RBF) approach which uses Voronoi tools in order to filter noise, reconstruct using different level of details and obtain a smaller final representation.

Recent improvements in automated shape acquisition have stimulated a profusion of surface reconstruction techniques over the past few years for computer graphics and reverse engineering applications. Data collected from scanning processes of physical objects are often provided as large point sets scattered on the surface object.

Functional based approaches where the surface is reconstructed as the zero-set of a function are standard. And the RBF approach has shown successful at reconstructing surfaces from point sets scattered on surfaces of arbitrary topology. The implicit function is defined as a linear combination of compactly supported radial basis functions.

We reduce the number of basis functions in order to obtain a more compact representation and to reduce the evaluation time. Reducing the number of basis function is equivalent to reduce the number of points (*centers*) where the functions are centered. Our aim consist in selecting a "little" set of relevant centers. In order to reduce the number of centers while maintaining decent fitting accuracy, we relax the one-to-one correspondence between the centers and the data points which is the rules in most of the RBF approaches. We depart from previous work by using as centers of basis functions a set of points located on an estimate of the medial axis. Those centers are selected among the vertices of the Voronoi diagram of the sample data points. Being a Voronoi vertex, each center is associated with a maximal empty ball. We use the radius of this ball to adapt the support of each radial basis function.

Our method can fit a user-defined budget of centers: the user can define the number of centers, i.e. the size of the representation and our algorithm will adapt the level of detail to this number using filtering and clustering or greedy selection.

Keywords

Reconstruction, Approximation, Interpolation, Regularization, Multiresolution, Implicit Surface, zero-Level sets, Radial basis functions, Voronoi, Medial axis, λ-Medial axis.

Résumé

Cette thèse s'inscrit dans la problématique de la reconstruction de surfaces à partir de nuages de points. Les récentes avancées faites dans le domaine de l'acquisition de formes 3D à l'aide de scanners donnent lieu à de nouveaux besoins en termes d'algorithmes de reconstruction. Il faut être capable de traiter de grands nuages de points bruités tout en donnant une représentation compacte de la surface reconstruite.

La surface est reconstruite comme le niveau zéro d'une fonction. Représenter une surface implicitement en utilisant des fonctions de base radiales (Radial Basis Functions) est devenu une approche standard ces dix dernières années. Une problématique intéressante est la réduction du nombre de fonctions de base pour obtenir une représentation la plus compacte possible et réduire les temps d'évaluation.

Réduire le nombre de fonctions de base revient à réduire le nombre de points (centres) sur lesquels elles sont centrées. L'objectif que l'on s'est fixé consiste à sélectionner un "petit" ensemble de centres, les plus pertinents possible. Pour réduire le nombre de centres tout en gardant un maximum d'information, nous nous sommes affranchis de la correspondance entre centres des fonctions et points de donnée, qui est imposée dans la quasi-totalité des approches RBF. Au contraire, nous avons décidé de placer les centres sur l'axe médian de l'ensemble des points de donnée et de montrer que ce choix était approprié.

Pour cela, nous avons utilisé les outils donnés par la géométrie algorithmique et approximé l'axe médian par un sous-ensemble des sommets du diagramme de Voronoi des points de donnée. Nous avons aussi proposé deux approches différentes qui échantillonnent de manière appropriée l'axe médian pour adapter le niveau de détail de la surface reconstruite au budget de centres alloué par l'utilisateur.

Mots-clés

Reconstruction, Approximation, Interpolation, Régularisation, Multirésolution, Surface implicite, Ensemble de niveaux zéro, Base de fonctions radiales, Axe médian, Voronoi, λ-Medial axis.

Remerciements

Je remercie xxxxxxxxxxxxxxxxxx, yyyyyyyyyyyyyyyyyyy et zzzzzzzzzzzzzzzzzzz

Abbréviations

Voronoi Background

- MA : Medial Axis
- MAT : Medial Axis Transform
- LFS : Local Feature Size

Reconstruction

- LOD : Level Of Details
- PCA : Principal Component Analysis
- MST : Minimum Spanning Tree
- MLS : Moving Least Square
- RBF : Radial Basis Function
- CSRBF : Compactly Supported RBF
- GRBF : Generalized RBF
- FMM : Fast Multipole Method
- PDE : Partial Differential Equations

Part I

General Introduction

0.1 Motivations

Theoretical and practical advances in signal acquisition and processing explain the rapid development of multimedia applications and the evolution of the informations manipulated: sound, images, videos and now 3D geometric models. The 3D models, by adding one dimension to the signal, allow to represent the reality or to (re-)invent it. 3D geometric models bring in addition to images and videos several specificities like interactivity, advanced rendering, viewpoint management. 3D models are not yet part of the mass market, although many applications benefit directly or indirectly to a large audience:

- *Medical applications*: computer aided diagnostic, therapy and surgery planning and monitoring require geometric physical modeling of organs and tissues of the human body. The current methods commonly resort to 3D images in order to extract a geometric description of the organs and lesions. Some example issues are the management of the lesions, that is the identification, characterization, reporting, storage and follow-up. An example: tumor detection in the brain (Fig.1:top right) needs a segmentation of the ill area and a geometric characterization such volume and growth speed in order to anticipate over the disease evolution and to plan surgery.

- *Engineering*: computer aided design (CAD) and simulation, which replace physical prototypes and experiences by geometric models and numerical computations, have been shown to considerably increase the productivity of the engineers. One example is the simulation of physical properties during car crashes in order to avoid real crash tests. An other example is numerical fluid simulation of an aircraft wing. We can distinguish forward from reverse engineering. In forward engineering a 3D model is created with CAD modeling tools starting from the sketch of an artist or from a list of requirements elaborated by the engineers. There is therefore no need for surface reconstruction in this case. In reverse engineering, the engineers start from an existing physical shape which comes either from physical modeling (e.g. a clay sculpture made by an artist), from an existing industrial product for which there is no access to the physical model (e.g. the product has been made a long time ago, or has been made by a competitor), or from a manufacturing process. In the latter case the goal is to measure the manufactured shape to check if it meets the initial tolerances and therefore the quality standards.

- *Cultural heritage*: from researchers to end users, applications using 3D models in the history and art field are developed. The creation of virtual museums allows the art and culture diffusion in the entire world. On an other side, the art digitalization is a powerful tool for the historians. This allows long time preservation but also virtual restoration or a certain understanding of the artist work/technique, for example by detecting the artist gesture. The *digital Michelangelo project* works on the creation of a long-term digital archive of some important cultural artifacts (such as the David of Michelangelo (fig.1:middle left));

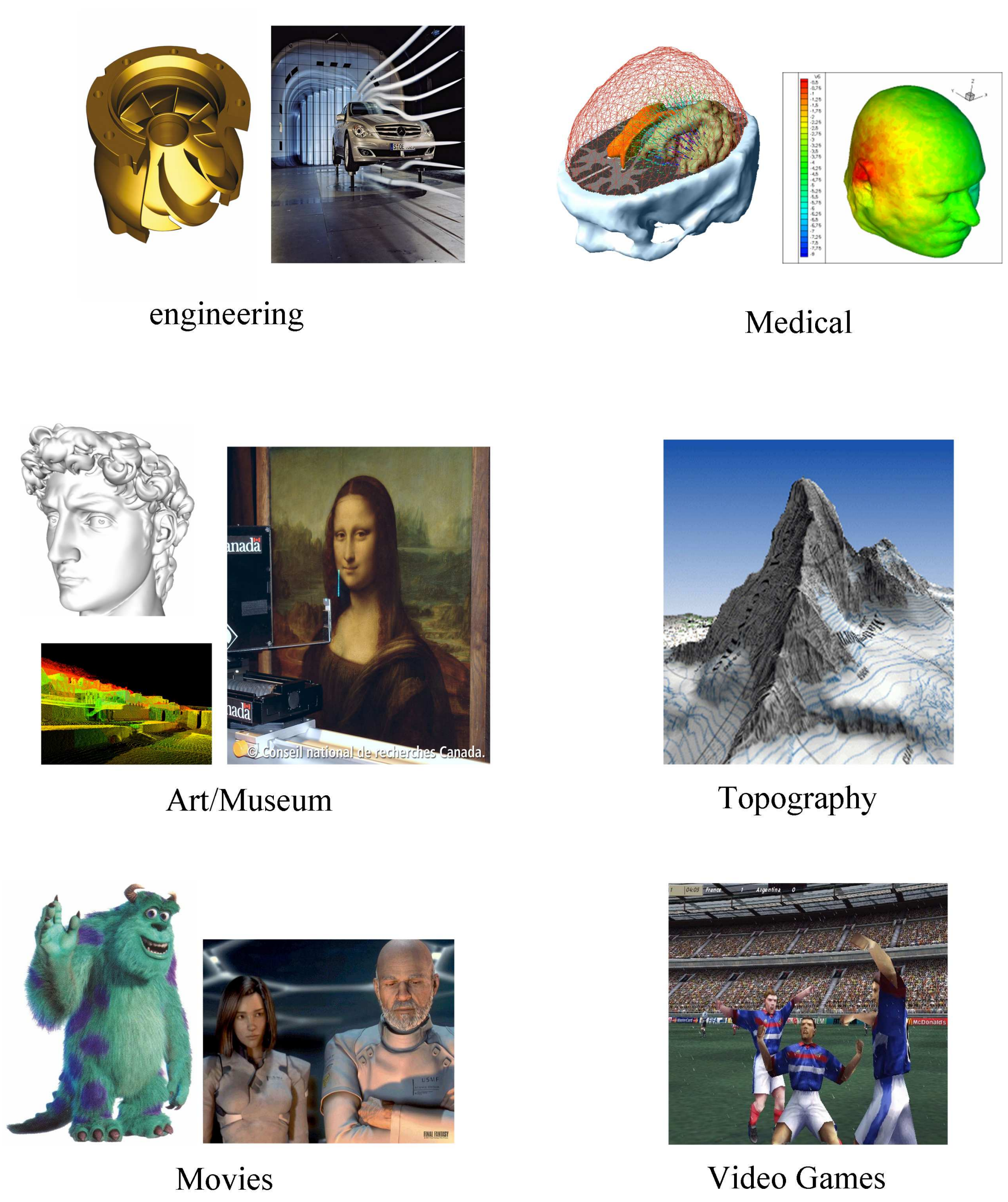


Figure 1: Application examples.

- *Video games*: one of the most inventive industries in 3D geometric modeling/processing. Video games seek for realistic or expressive rendering and for rapid interactivity with the user (frame rate is critical). Thus a lot of work is done on 3D models in order to find realism of the shape, gestures, character feelings. For these issues, it is often simpler to request for surface reconstruction than to reproduce the reality from scratch. This is even more true for animated characters for which the poses and gestures are notoriously difficult to reproduce. For articulated objects, a solution is to model the skeleton of the object and then to animate each part of the skeleton. An other solution is to perform *motion capture*. Motion capture involves measuring an object position and orientation in physical space by using sensors. Then the motion is reproduced on a 3D model. The objects of interest include human and non-human bodies, facial expressions,etc.

- *Movies*: for the movie industry time has come where 3D modeling is easier and cheaper for some photorealistic scenes, as well as for special effects. In terms of needs for reconstruction there are a lot of acquisition methods like camera tracking. Furthermore, using 3D animated models make possible to prototype, in real time, the different scenes of a movie and thus to create a 3D story board.

- *Topography*: The growing popularity of GPS-driven navigation systems have rekindled the interested in the accelerated 3D modeling of large scale environments, like cities. An other application of 3D model is in the optimized land resource management. 3D model allow to work on reliable and detailed information describing the spatial distribution of soils, geology, topography. The data are acquired from satellite image, from sonar. In addition to GPS and soil studies, some users need 3D geographic informations for urbanism, army, telecommunications and urban transport. The visualization is important but also the possibility to perform simulation: earth movements, air or sea current, for example.

Although all applications mentioned above require specific processes, they can be classified by their final goal which is pure visualization, simulation or calculation. One common trend between these three classes of applications is the ever increasing need for accuracy, be it for high definition realistic rendering or for accurate computation and simulations.

An object can be defined in several levels: semantic (abstract), mathematical or digital. On a computer, the object representation must be finite thus a discretization must be performed in order to convert a real object to its digital representation.

In this thesis we investigate the topic of surface reconstruction from data acquired onto a physical object.

Depending on the techniques used, the output of a scanning process can be simply a set of points (unstructured data), but it can be also a profile, a range image or a volumetric output (structured data). The standard acquisition techniques can be roughly divided into two main categories: acquisition by contact or acquisition without contact. In the first case, the shape is acquired by touching the object surface on each relevant side with an ad-hoc instrument. We are interested by the second class of techniques where the shape is acquired by indirect techniques based on a couple source/sensor. The energy is emitted by the source and the sensor measure the return of the signal. Generally, the acquisition system is an active optical system composed of a laser source and a sensor (fig.2) [RCM+01]. The source emits a certain illuminant pattern and the sensor acquires returned marks reflected by the

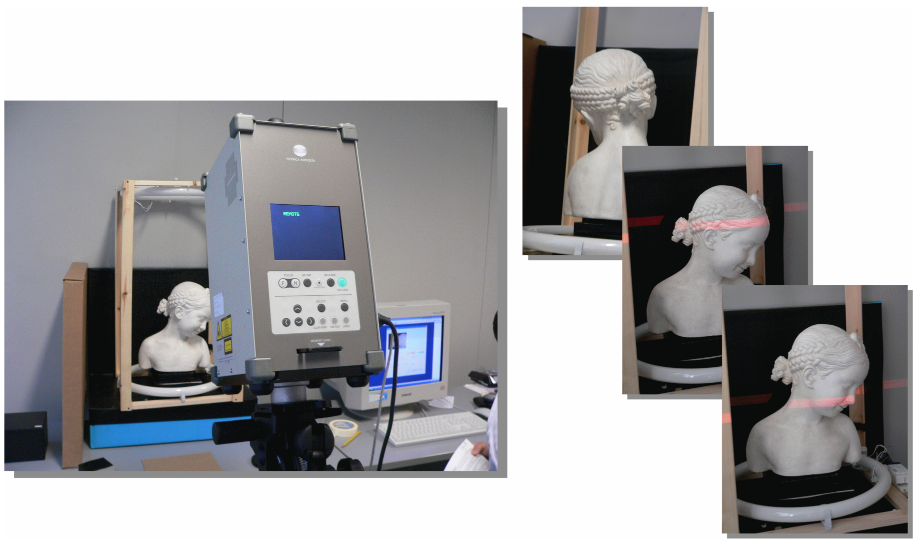


Figure 2: Bimba con nastrino. Sculpture digitized by a Minolta V910 laser scanner.

object surface. The source scans regularly the space and the system returns a 2D matrix possibly sparse, called *range image*, identifying points on the surface. More precisely, a range image is a list of 3D coordinates in a given reference frame, i.e a point cloud, for which no specific order is required.

Scanning an object consists of set of complex tasks which are commonly referred to as the *3D scanning pipeline* (Fig.3).

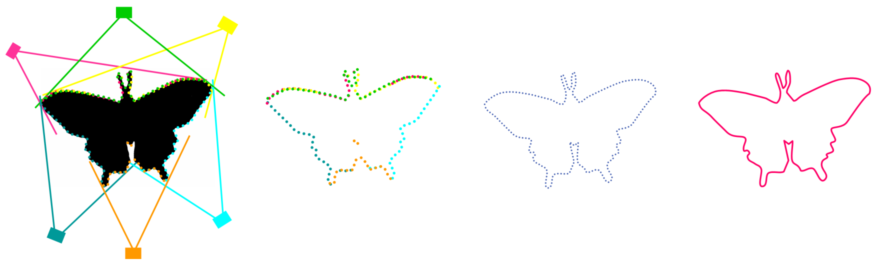


Figure 3: Scanning pipeline. Left to right: Acquisition (the physical object is acquired by several scanning process in order to cover all the surface), registration and merging.

- *Acquisition* of range images (Fig.2). One single view is not enough to reconstruct the whole surface due to occlusions, shadows,etc. Thus the main issue is to decide on the set of range images: number, position, specifications, resolution, so as to obtain a complete sampling of the surface. The point set should cover the whole surface with no holes and densely enough with respect to the local feature size. Note that a partial overlapping is necessary for the next step in order to find common features in two successive range images.

- *Registration* of range maps (Fig.4). Each range image is taken independently, hence defined in a local coordinate frame which is relative to the current sensor location.

The registration is the process to find the rigid transformations (translations, rotations) to apply to each range image in order to register all the points in a single coordinate frame.
Thus, correspondence between the different range image must be found, that is several common feature points must be detected either automatically or manually. For automatically registering two unstructured 3D point sets, ranges images, the classical approach is to perform an ICP (Iterative Closest Point). Generally, the ICP algorithm is performed for all pairs of successive range images and then for all the range images together.
After registration, some parts of the range images which correspond to the same surface area mildly overlap.

- *Merging.* The issue is to build a single, non redundant surface out of the many, partially overlapping range images. That is, to reconstruct the surface of the scanned object.

- *Processing.* The quality of the reconstructed mesh can be improved by a series of tasks commonly referred to as geometry processing: denoising, smoothing and fairing. In some cases it may also be desirable to edit the mesh manually so as to perform some modification (deformations, blending,...);

- *Simplification.* The reconstructed mesh is often overly complex and there may be a certain redundancy in the vertices with respect to the physical shape digitized, or for the targeted application. The mesh can be simplified by decimation or remeshing so as to reach a user-defined complexity expressed in number of primitives.

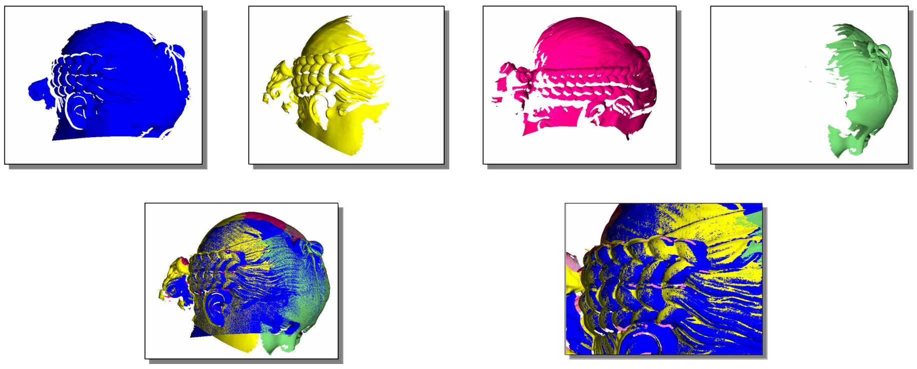


Figure 4: Registration. Top: Four range images of the Bimba con nastrino (18 range images were necessary to scan the sculpture). Bottom left: the range images are merged. Bottom right: detail of the braid.

0.2 Goals

Our work takes place in the *merging* step aforementioned. From the point sets sampled by the scanner and registered, our aim is to reconstruct a discrete representation of the object surface. This issue is known as the surface reconstruction problem in the literature.

Recent improvements in automated shape acquisition have stimulated a profusion of surface reconstruction techniques over the past few years for computer graphics and reverse engineering applications. Data collected from scanning (fig.2) processes of physical objects are often provided as large point sets scattered on the surface object.

The main difficulties encountered during surface reconstruction may come from two sources: the shape of the object or the data acquired. On the one hand, the shape may have a non trivial topology or sharp features. On the other hand, the data sampled may be noisy, the sampling may be non adapted and the point set may be large, if not massive.

Noise: The scanning pipeline is entangled with noise, which translates into an uncertainty over the location of points and even over their physical existence. All sources of noise are not known, therefore it is hard to get a precise knowledge of the nature of the noise. Generally, the noise may come from two sources:

- *Acquisition*: There is uncertainty from the physical measure, as the sensor involves physical and electronic devices. All electronic devices suffer from *electronic noise* to a greater or lesser extent. In scanners, this noise has its greatest effect in low light level detection, i.e. while scanning the dark areas of objects. The optical devices (laser, lenses) are also sources of noise due to their uncertainty in their properties (wavelength, geometry and material of lenses).

  Other noise may come from the material of the scanned object. When a laser beam enters a marble block, for example, it creates a volume of scattered light whose apparent centroid is below the marble surface. The reflected spot seen by the range camera is shifted away from the laser source. Since most laser triangulation scanners operate by detecting the center of this spot, the shift causes a systematic bias in derived depth.

- *Registration*: range images may not coincide for noise or sampling reasons. Thus the alignment may be wrong or simply inacurrate and then produces a set of different layers where the range images overlap.

Moreover, selecting the n viewpoints is not easy: the overlapping rate of the range images must be sufficient but not too redundant. For example, Some surface regions can not be captured: they are visible from the emitter but not by the receiver and vice versa (fig.5). Thus the user selection of mildly overlapping patches is non trivial, In most of the cases, a first set of range images is measured which allow the user to decide if others scans are necessary.

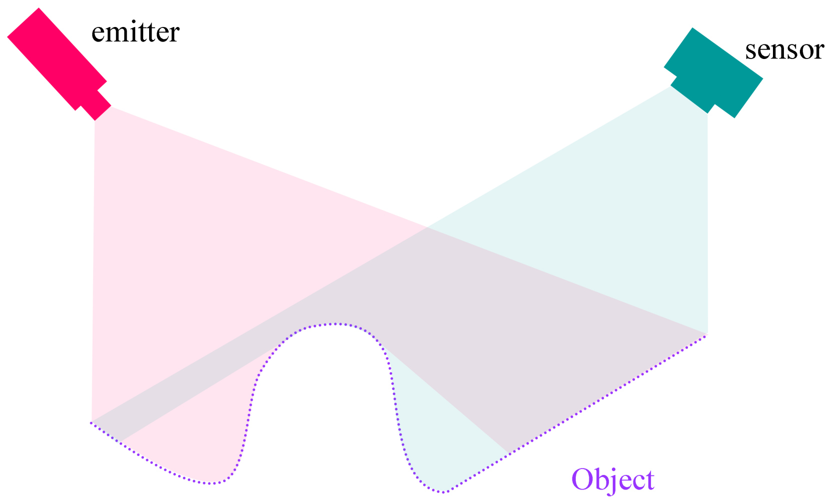


Figure 5: One of acquisition problem: hiding area. The visibility area of the emitter (pink) and the sensor/receiver (green). Some surface regions may be visible from the emitter and not visible from the receiver and vice versa.

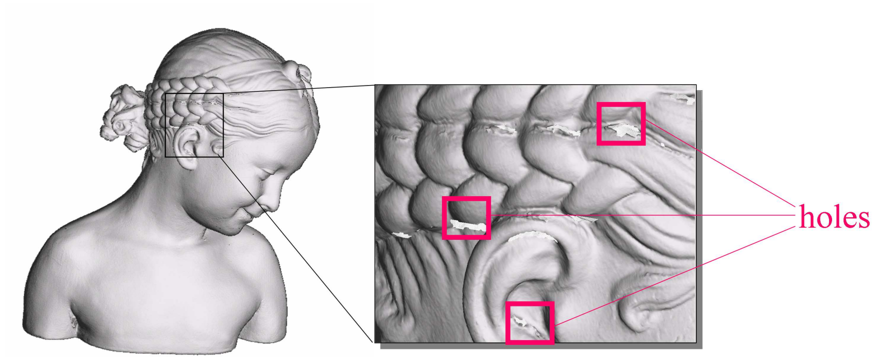


Figure 6: Reconstructing the Bimba con nastrino. The registration and fusion stages have been made with Minolta's software). Some holes appear (right) where the scan can not capture the shape (deep area)

Non adapted sampling: One important issue is to obtain a good sampling. A good sampling is a point set which density locally adapts to the local geometry of the surface. Beside the limited accuracy of the scanner, the main difficulty come from the facts that the acquisition process is performed manually, and that the user has no a priori knowledge over the shape except from visual inspection. Some parts of the surface object can therefore be over-sampled (areas with a large overlapping rate of the pair of successive range images) whereas some other parts may be under-sampled (hidden area or the surface region tangent to the laser beam) (see Fig.6).

0.3 Contributions

The contribution of this thesis is the development of a surface reconstruction method based on the Radial Basis Functions (RBF) approach. We use Voronoi tools in order to filter noise, reconstruct using different level of details and obtain a smaller final representation.

Among many techniques devoted to surface reconstruction, functional based approaches where the surface is reconstructed as the zero level set of a function are highly popular. The reconstruction process amounts to search for a function whose zero level set passes though or near the data points. The implicit function is defined as a linear combination of compactly supported radial basis functions. We depart from previous work by using as centers of basis functions a set of points located on an estimate of the medial axis, instead of the input data points. Those centers are selected among the vertices of the Voronoi diagram of the sample data points. Being a Voronoi vertex, each center is associated with a maximal empty ball. We use the radius of this ball to adapt the support of each radial basis function.

Our method can fit a user-defined budget of centers in two ways. In the first case, the selected subset of Voronoi vertices is filtered using the notion of lambda medial axis, then clustered to fit the allocated budget. In the second case, the set of centers is selected among the Voronoi vertices with a greedy algorithm.

The combination of radial basis functions and Voronoi-based surface reconstruction allows us to reconstruct a smooth and watertight surface by approximating the signed distance to the sampled surface defined all around the sampled shape. Furthermore, our choice for the centers allows reducing the center number to obtain a more compact representation in term of centers, coefficients and supports. The user can define the number of centers, i.e. the size of the representation and our algorithm will adapt the level of detail to this number using filtering and clustering, or greedy selection.

0.4 Outline

This thesis is organized as follow, the first part outlines the theoretical framework underlying surface reconstruction and presents a state of the art in surface reconstruction from point set. The second part is devoted to present our contributions.

Part II

State of the Art: Reconstruction from Points

Chapter 1

Introduction

Definition 1.1. *A **surface** $S \subset \mathbb{R}^3$ is a 2-manifold embedded in $\mathbb{R}^3$. Each point $p \in S$ has a neighborhood homeomorphic to an open disk or to an open halfdisk of $\mathbb{R}^2$. The points with neighborhoods homeomorphic to an open halfdisk constitute the boundary of S.*

We first define the main classes of surface representations. In the following, we consider oriented manifold surfaces. These surfaces divide the space into two subspace: a bounded volume tagged as inside and an unbounded volume tagged as outside.

1.1 Surface Representations

In computer graphics a large variety of geometric representations has been used for reconstruction, modeling, editing and rendering of 3D objects.

We can define a **surface representation** as a data structure which allows performing various operations:

- **visualization** of the surface;
- **query**: determine if a given point is inside or outside the surface, compute the distance to the surface,...;
- **modification**: a surface can be deformed, blended with an other surface or animated;
- **evaluation** of differential quantities at a given point on the surface (first derivative, curvature,...).

The required properties of a given surface representation may vary according to the targeted applications: from ease of visualization to data structure suitable to geometry processing, through ease of modification and animation.

We define two main classes of surface representations: explicit and implicit. An explicit formulation describes the precise location of the surface as a set of primitives or functions. An implicit definition represents the surface as an isocontour of a scalar function.

1.1.1 Explicit Representation

A 3D model can be defined by a collection of primitives such as points and triangles. By adding connectivity relationships between the primitives we may obtain a mesh. When the surface patches are spline surfaces, the surface representation is parametric.

Most explicit surface representations share common properties: In general the surface can be rendered efficiently but it is difficult to perform certain geometric operations such as determining if a given point is inside or outside the surface, or blending together two surfaces.

We now describe three of the main explicit surface representations with their strengths and weaknesses.

Remarks: terrain modeling is a particular case of explicit surface. The surface is represented as the graph of a function of two variables $f : \mathbb{R}^2 \rightarrow \mathbb{R}$. This definition fails to represent all the surfaces (like watertight). Thus we can not summary explicit surfaces as terrains.

1.1.1.1 Collections of Primitives

A point-based surface representation is a sampling of a surface, resulting in 3D positions, optionally with associated normal vectors or auxiliary surface properties such as color or physical attributes.

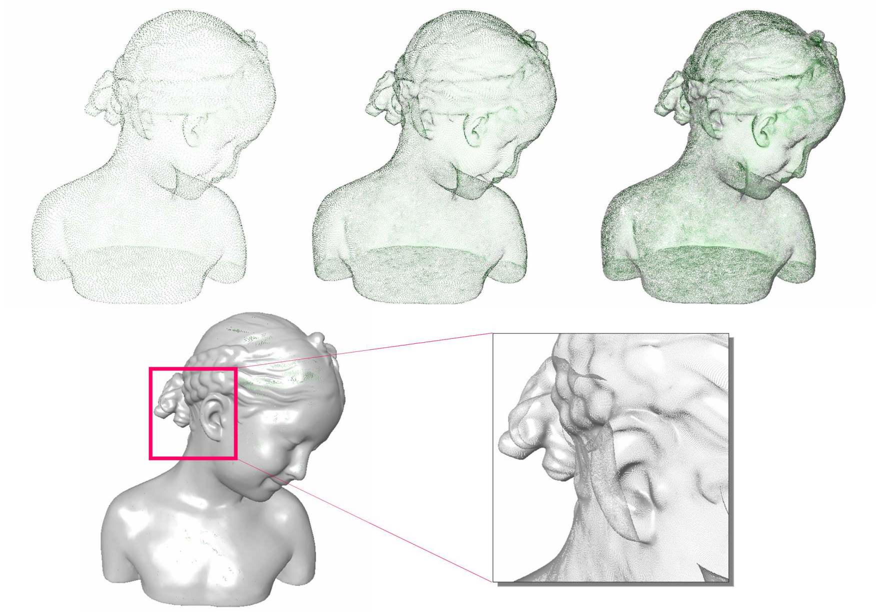


Figure 1.1: Point sample representation. Three different point sampling densities for the Bimba model. Top: three point sets: 25K, 74K and 640K points. Bottom: a point set with 2013K points and a closeup of the ear.

The point set must be dense enough to faithfully represent the shape (fig.1.1). The point density must be adapted in order to obtain a compact representation and avoid redundancy.

Surface splats have been proposed in order to bridge the gaps between neighboring point samples. The points are enriched with normal vectors and a radii, turning them into object-space circular disks. A locally optimal adaptation to the curvature of the underlying surface is provided by elliptical splats (fig.1.2). The latter are defined by two tangential axes and their respective radii. Optimal local approximation is achieved if the two axes are aligned to the principal curvature directions of the underlying surface and the radii are inversely proportional to the corresponding minimum and maximum curvatures.

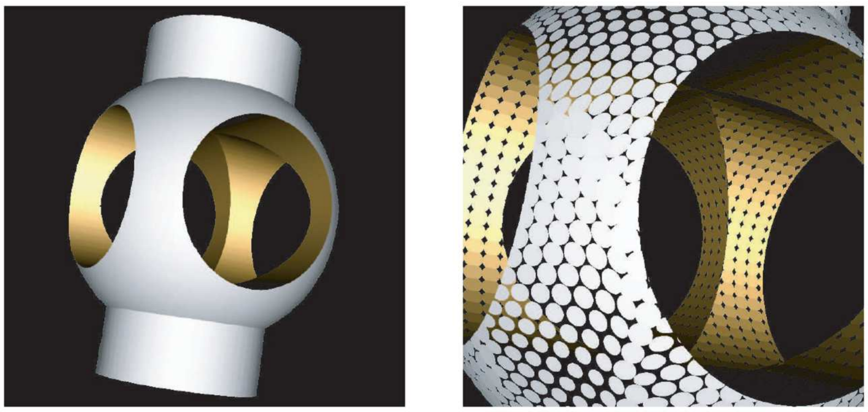


Figure 1.2: Surface splats (image taken from [?]).

Despite its simplicity, the splat surface representation requires a dense sampling even in smooth regions: if splats are small compared to spacing then gaps result.

1.1.1.2 Meshes

A mesh is composed by a geometry and a connectivity, respectively a collection of primitives and a set of relationships between these primitives (fig.1.3).

Triangle meshes (fig.1.3:left) are the most common surface representations in many applications due to their simplicity and flexibility. More formally, the surface is defined as a simplicial complex with vertices, edges and facets.

Definition 1.2. *A 3-dimensional simplicial complex is a collection of simplices of dimension at most 3 in $\mathbb{R}^3$ such that all simplices spanned by vertices of the complex belong to the complex and the intersection of any two simplices is either empty or is a simplex belonging to both simplices.*

When the facets differ from triangles as polygons with arbitrary degrees the mesh is polygonal (fig.1.3:left).

1.1.1.3 Parametric Surfaces

Parametric representations such as spline surfaces [Far02] are defined as functions mapping planar domains Ω to $\mathbb{R}^3$.

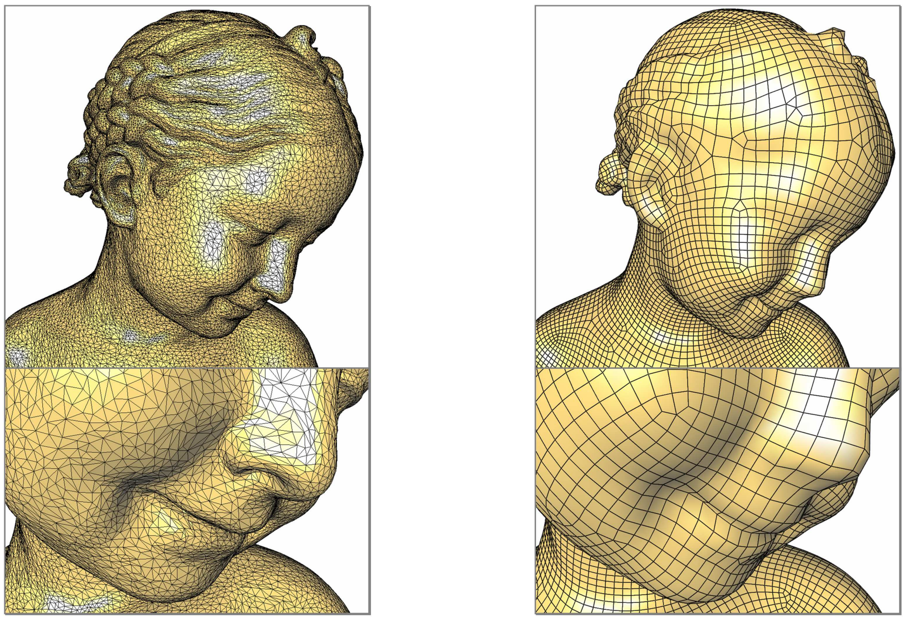


Figure 1.3: Three different meshes of the Bimba. Left: Triangle mesh. Right: Quadrangle mesh.

Definition 1.3. *For **parametric representation**, the surface is represented by a two dimensional function* $f : \mathbb{R}^2 \to \mathbb{R}^3$ *which maps a two dimensional parameter domain* Ω *into a three dimensional space. Surfaces are represented as :*

$$f(u,v) = (x(u,v), y(u,v), z(u,v)). \tag{1.1}$$

As the function f is a homeomorphism from Ω to the surface, several kinds of surface, such as the ones with handles, may not be represented by a single parametric function. Theses surfaces are represented by a set of parametric surface patches which are stitched together with geometric continuity conditions [Far02]. Examples of such parametric representations include B-splines, Bézier surfaces, Coons patches and non-uniform rational B-splines (NURBS).

1.1.2 Implicit Representations

The surface is defined as an isocontour of a scalar function $\mathbb{R}3$ $f : \mathbb{R}^3 \to \mathbb{R}$, i.e. the zero level set of f (fig.1.4).

Definition 1.4. *The surface* S' *is represented implicitly as the zero-set of a function* f*, i.e*

$$S = \left\{p \in \mathbb{R}^3, f(p) = 0\right\} \tag{1.2}$$

Note that a surface is this way non uniquely determined as the isocontour of several functions may correspond to the same surface. Knowing a surface, a common choice for the function f is the signed distance function to the surface.

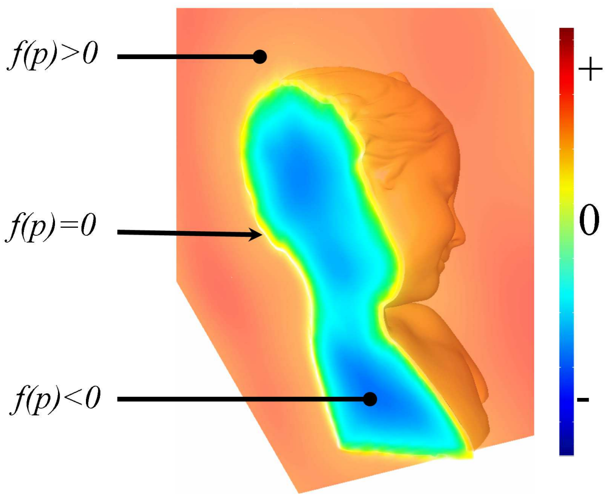


Figure 1.4: Implicit representation. The Bimba surface (gold) is defined as the zero-set of a function f, positive outside and negative inside. The colors on the cutting plane represent the function values (cold tones for negative values, warm tones for positive values and white for zero values).

1.1.3 Comparison

Location query Point location queries are easier for implicit representations than for for explicit representations. Assuming the surface being defined as the zero-set of a function f, a point p can be located by a simple evaluation of f at p: p is inside if $f(p) < 0$ and outside if $f(p) > 0$ (fig.1.4). Point location for an explicit representation such as a triangle mesh without boundary is more difficult, as it requires to know a point q inside the surface (or outside) and a relevant data structure in order to check if the segment pq intersects the surface.

Visualization Explicit surface representations are in general easy to visualize, as it amounts to iterate and render over all primitives or surface patches. Conversely, implicit surface representations are considerably harder to render as it requires a discretization step in order to generate a set of simple primitives. This in fact amounts to convert the implicit representation into an explicit one, restricted to one or several user-defined isovalues. The isovalues of the implicit function are commonly discretized into triangles using surface extraction algorithms like the *marching cubes* [LC87, Blo94] or using meshing technique such as the Delaunay-based surface mesher elaborated by Rineau et al [**?**].

Texture mapping may be required for the visualization process. This operation is not trivial and needs a parametrization of the surface. Thus, the parametric representation is the only one which allows direct texture mapping.

Modification Modelisation and animation of 3D models require a set of operations such as rigid transformations, deformations, blending and Boolean operations. The main surface representations listed above mainly differ by the type of control available over the geometry and topology for each operation.

Mesh and parametric representations provide control over the topology of the object. For example, during local distortion of a mesh, the connectivity can be updated while maintaining strict consistency conditions to preserve a manifold surface structure. On the other side, topology modification, like adding or removing handle, may be difficult. For parametric representation the topology can be controlled explicitly. However a modification of the object such as deformation or topology change can require modifications of the domain Ω in order to avoid strong distortion and inconsistencies.

Finally, while implicit surface representations can represent surfaces with arbitrary topology, it is hard to predict the topology changes during deformation. Boolean surface operations are simpler for implicit than for explicit surface representations.

Evaluation Computing differential quantities at a given point is notoriously difficult for surfaces defined by a collection of primitives or by a mesh [**?**]. Conversely when a surface is defined by functions, be they parametric or implicit, the tangents, the normal and the curvatures can be evaluated at any point.

1.2 Surface Reconstruction

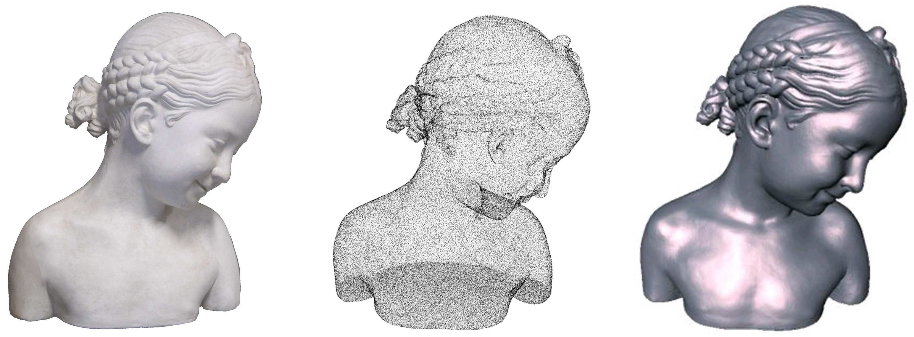


Figure 1.5: Reconstruction pipeline. The Bimba sculpture (left) is scanned to obtain a point set, P, scattered on the surface object (middle). Then the surface, S, of the sculpture is reconstructed (right) (a surface S' is obtained).

The input of a the reconstruction algorithm is a point set $P = \{p_i\}_{i=1..n}$ measured on the surface S either manually or via a physical process such as 3D scanning (fig.2). We assume that the original surface S is smooth and that the sampling is dense enough, especially near features such as edges, points and bumps. The output of the reconstruction algorithm is a representation S' of the surface S.

The reconstructed surface S' may interpolate or approximate the data points P (fig.1.6).

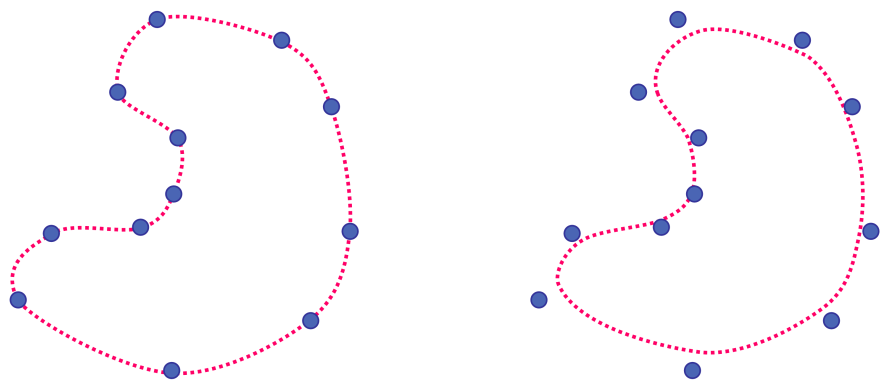


Figure 1.6: Curve reconstruction. Left : the data points are interpolated by the blue curve. Right : the data points are approximated by the blue curve.

In the interpolating case, the surface S' must pass through all data points P. Note that the solution is not unique (Fig.1.7) it depend on the approach chosen.

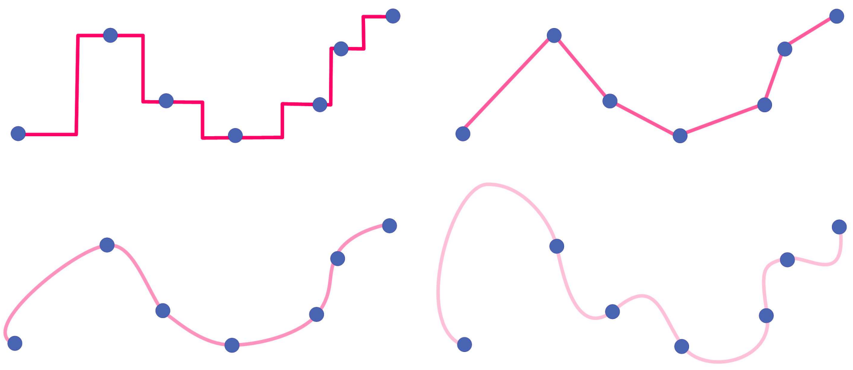


Figure 1.7: Different curves which interpolate the data points (piecewise constant, linear,.. interpolation)

Conversely, in the approximating case, the surface S' must pass close but not necessarily through the data points P.

For both cases the surface reconstruction problem is inherently an ill-posed problem, as an infinite number of surfaces could satisfy the constraints listed above. A common idea to reconstruct the most plausible surface is to infer geometric or physical properties for the measured surface. For example, geometric properties ensure that the reconstructed surface has to be smooth. Physical properties ensure that the surface has a minimum curvature by minimizing during reconstruction an energy functional.

Reconstruction methods can be roughly classified into two main classes : Delaunay based (sec.2.1) or functional based (sec.2.2).

The mesh based reconstruction algorithms establish connections between samples which are neighbors on the surface. These approaches used on geometric constructions which defines a simplicial complex on these samples, typically the Delaunay triangulation or its dual the Voronoi diagram.

In the functional based approaches, the approximated surface S' is formulated implicitly as the zero level set of a function f defined all over the space or locally. In most cases, the computed function f is an approximation of the signed distance function to the surface S (see [TO02] for a survey).

It remains several important approaches which may not be classified as Delaunay based or functional based. These approaches consist in finding the min/max cut of a graph; or are based on statistical measures or involve deformable models.

The difficulty of the reconstruction process is to deal with non smooth surfaces, with noisy data,... The aim is to obtain a watertight surface, with a compact representation and to have an algorithm with the fewest parameters.

Chapter 2

Surface Reconstruction from Points

In the following, we restrict ourselves to the reconstruction techniques which take as input a set of unorganised points $P = \{p_i\}_{i=1..n}$ assumed to lie on or near the surface S of an unknown object. The result of the reconstruction algorithm is a surface S' that approximates S. The representation (Chap.1.1) used for S' depend on the reconstruction method chose.

2.1 Delaunay Based Surface Reconstruction

A popular approach is to reconstruct a surface using a Delaunay triangulation of the input point set or using its dual the Voronoi diagram. The main idea is the following: when the surface is sampled densely enough, the points which are closed in 3D should be closed on the surface. Therefore the Delaunay triangulation, which encodes the Euclidean distance between the sample points in 3D is the tool of choice for establishing their neighborhood relationships on the surface.

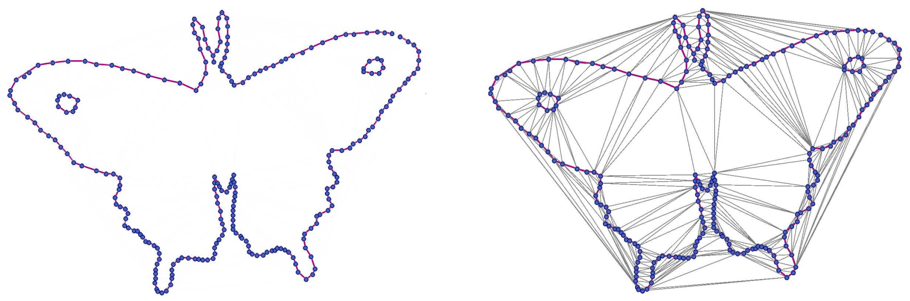


Figure 2.1: Delaunay based curve approximation. Left: the blue points are sampled on a red curve. Right:the Delaunay triangulation of the point set contains a piece-wise linear approximation of the curve

In general the Delaunay-based approaches sculpt the Delaunay triangulation of the sample points. More precisely, a subcomplex interpolating the sampled surface is extracted from the Delaunay triangulation by greedily eliminating facets from the tetrahedra according to geometric criteria such as the area of the triangular facets (fig.2.1, see [?] for a survey).

2.1.1 α-Shapes

The α-shapes have been introduced by Edelsbrunner's and Mücke's in 1994 [EM94]. Given a finite point set P, and a real parameter α, the α-shape of P is a polyhedral surface which is not necessarily connected (fig.2.2). The set of real numbers α leads to a family of α-shapes.

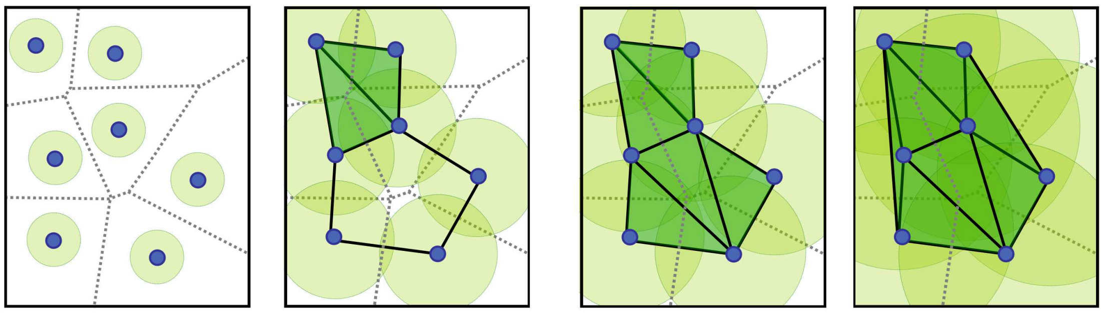


Figure 2.2: Reconstruction using α-shapes. The input points (blue) with α-balls centered on them. From left to right: the parameter α is increasing. The black edges compose the α-complex and the α-shape is the boundary of the green area.

Definition 2.1. *Given* $0 \leq \alpha \leq \infty$. *Let a* α-***ball*** *be an open ball with radius* α. *A certain* α*-ball* $\mathcal{B}_\alpha$ *is called empty if* $\mathcal{B}_\alpha \cap P = \emptyset$.

Remark: A 0-ball is a point. An ∞-ball is an open half-space.

Definition 2.2. *The* α-***complex*** *is the Delaunay triangulation restricted to* α*-ball. The points* $p, q \in P$ *are connected by a straight edge if there is a ball* $\mathcal{B}_\alpha$ *that passes through* p *and* q*, and all other point of* P *lie strictly outside* $\mathcal{B}_\alpha$.

In other words, given a finite set of points P and α, the α-**complex** of P is the dual complex of the Voronoi diagram of P restricted to the union of balls $\mathcal{B}_\alpha(P)$. Each restricted Voronoi cell is the intersection of the original Voronoi cell with the ball of its generating point.

By construction the α-complex is a sub-complex of the Delaunay complex for every $\alpha \leq 0$. With increasing α more and more cells of the Delaunay complex appear in the α-complex (fig.2.2). A family of shapes can be derived from a Delaunay triangulation of the point set P; where the parameter α controls the level of detail. The 3D α-**shape** is defined as the union of the cells of the α-complex. The α-shape degenerates to the point-set P when $\alpha \to 0$. On the other hand, the α-shape for $\alpha \to \infty$ is the convex hull of S.

The α-shapes method consists of three steps: first, a triangulation of the point set is computed, then the α radius is selected and at last, the simplexes that are to be included in the reconstructed shape are identified.

This approach allows fast, accurate, and efficient calculations of volume and surface area. One drawback is that the sampling needs to be more or less uniform.

α-shapes are based on an underlying triangulation that may be a Delaunay triangulation in case of basic α-shapes or on a regular triangulation in case of weighted α shapes. The α-shapes can be extended to deal with weights [?] by using a pseudo distance measure. The power distance is defined as the square of the Euclidean distance minus the weights. Let p_1 and p_2 two points with weight w_1 ans w_2:

$$d((p_1, w_1), (p_2, w_2)) = \|p_1 - p_2\|^2 - w_1 - w_2 \tag{2.1}$$

The power distance is zero if and only if two spheres are orthogonal.

2.1.2 Crust

The Crust algorithm, also called Voronoi filtering, for surface reconstruction was designed by Amenta and Bern [AB98].

This algorithm relies the notion of the **poles** and **medial axis** (see Annex9).

For the reconstruction problem, points are measured on a surface of an input shape. In 2D, if the sampling density of the shape goes to infinity, the vertices of the 2D Voronoi diagram approach the medial axis (see proof in [?]). However, a similar result does not hold in 3D. Some Voronoi vertices may lie very close to the surface and thus far from the medial axis.

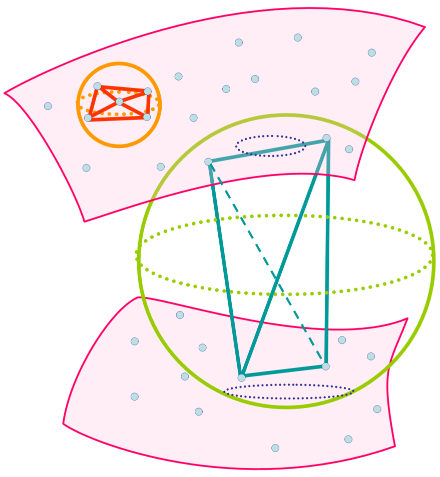


Figure 2.3: Tetrahedron configurations in 3D. The orange tetrahedron corresponds to a Voronoi vertex far from the medial axis. The green tetrahedron corresponds to a Voronoi vertex near the medial axis, i.e. a pole. The poles are guaranteed to converge to the medial axis.

In [AB98], Amenta and Bern observe that if some Voronoi vertices remain far from the medial axis, some other ones, so-called *poles*, lie close to the medial axis.

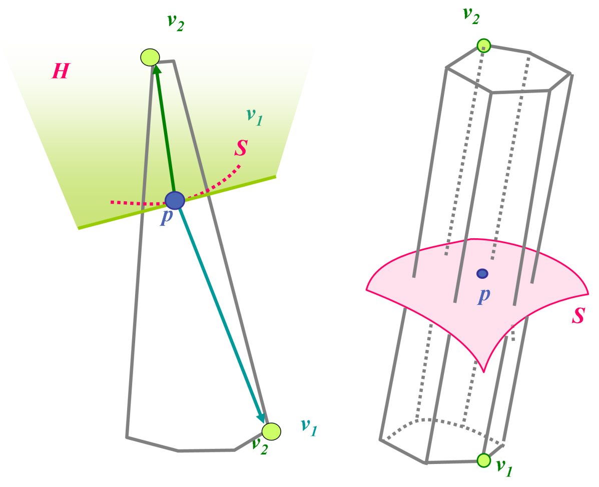


Figure 2.4: Pole. One point p, its Voronoi cell and 2 poles v_1 and v_2. Left: in 2D case. Right: in 3D case.

In the following we note $V_p = V_{P,p}$ the cell associated to p in the Voronoi diagram of the point of P.

Definition 2.3. *The set of* ***poles*** *is a subset of Voronoi vertices. At most 2 poles can be extracted for each Voronoi cell V_p, which means that for each point $p \in P$ correspond at most two poles. Let V_p be a bounded Voronoi cell. The first pole v_1 is the Voronoi vertex in V_p with the largest distance to the sample point p. The second pole is the Voronoi vertex v_2 in V_p furthest away from p in the opposite half space of v_1, i.e such that the vector $\overrightarrow{pv_1}$ and the vector $\overrightarrow{pv_2}$ make an angle larger than $\frac{\pi}{2}$ (Fig.2.4).*

Boissonnat and Cazals [?] and Amenta et al. [ACK01] show that under strict conditions, a smooth original surface and a dense sampling, the poles can lead to a good approximation of the medial axis (see proof in [AB98]). Besides, the vector vertex-pole provides a good approximation of the normal of the original surface (fig.2.4).

Algorithm: Assuming a certain sampling density on the surface, the algorithm consists in four steps :

- Compute Voronoi diagram of P;
- For each $p \in P$ find its poles (v^+, v^-), let $V(P)$ be the set of poles;
- Compute Delaunay triangulation T of $P \cup V(P)$;
- Return all faces in T with vertices in P as an approximation S' of the surface S

Besides an approximation of the surface, the algorithm provides an approximate medial axis of the S as the set of facets in T which are not in the approximate surface.

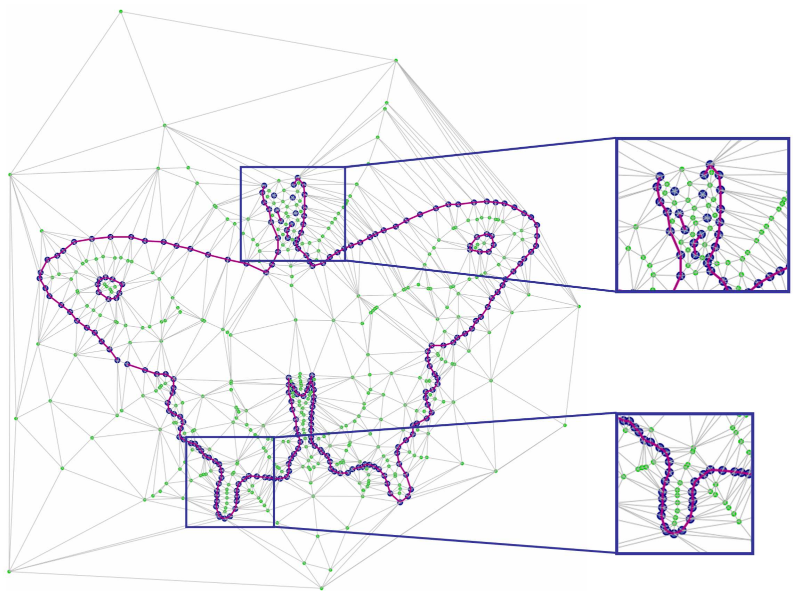


Figure 2.5: Crust in 2D. The blue input points P, the green Voronoi vertices $V(P)$, all the gray Delaunay edges and, highlighted, the Delaunay edges between points of P (red). Right: two details. Top closeup image, the antennas are under sampled: the reconstruction failed. Bottom close up the image, the spurs are well sampled: the shape is reconstructed.

The Crust algorithm was one of the first surface reconstruction algorithm to provide theoretical guarantees.

2.1.3 Power Crust

The *Power Crust* algorithm combines concepts of medial axis, Voronoi diagram, and power diagram. As the Crust it assumes that the sampling density is higher where there is more detail. It extends to handle noise, sharp corners, and non-watertight surfaces.

Algorithm :

1. Compute the Voronoi diagram of sample points;
2. Determine which Voronoi vertices are poles;
3. Compute the power diagram of the poles weighted by the radius of their polar ball;
4. Determine which poles are interior and exterior;
5. Return the union of the power diagram cells of the inside poles.

The algorithm provides an estimate of the interior medial axis. The power diagram to define the adjacencies of the polar ball centers, i.e. the poles. Subsets of inner (resp. outer) poles whose power diagram cells share a face are connected with a dual weighted Delaunay face. These faces form a simplicial complex, the power shape, analogous to the medial axis.

2.1.4 Cocone

The cocone algorithm was designed by Amenta et al [ACDL02] as an improvement over the Crust algorithm. The cocone algorithm computes a set of candidate triangles containing the restricted Delaunay triangulation $D_s(P)$. The cocone can be seen as the restricted Delaunay triangulation of the input points enlarged by co-cones.

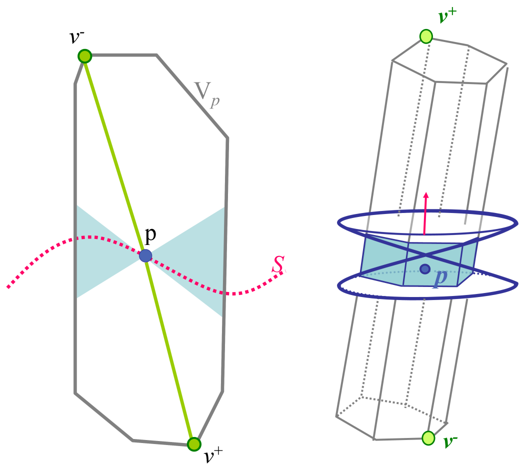


Figure 2.6: Co-cone in 2D and 3D. The co-cone of a sample point p on a curve together with the Voronoi cell of the sample point and its pole v^+.

Algorithm:

- Compute the Delaunay triangulation of P;
- For every sample point $p \in P$, approximate the normal of S at p using the pole of the Voronoi cell V_p in $V(P)$ (like in sec.2.1.2);
- For each sample point p select a set of candidate triangles.

Follows, let $p \in P$ a sample point. The co-cone at p is defined as the intersection of V_p, the Voronoi cell of p, with the complement of a double cone with apex p and fixed opening angle around the approximate normal at p.
A triangle t in $D(P)$ is **candidate** for p if its dual Voronoi edge e intersects the co-cones of p (fig 2.6). If the sampling density is sufficiently high, these candidate triangles lie close to the original surface S. A subsequent manifold extraction step extracts a manifold surface out of this set of candidate triangles.

The Cocone algorithm demands a single Delaunay triangulation in contrast to the Crust algorithm. We can notice some problems with practical data, due to undersampling, noise or non-smoothness. For example, the estimated normals may not be correct, leading to a wrong choice for the triangles.

Tight cocone [DG03] attempts to fill such holes by labeling Delaunay tetrahedra as in or out, based on the initial approximation of the surface. It removes all outside tetrahedra, and take the boundary of the inside tetrahedra to get a surface S'.

2.1.5 Flow Complex

The flow complex is a data structure that can be used to structure a finite set of points in $\mathbb{R}^3$. The flow complex is closely related to the Delaunay triangulation, but neither complex is a subcomplex of the other. The striking difference is that it seems much easier to extract a surface or cavity model from the flow complex than from the Delaunay triangulation.

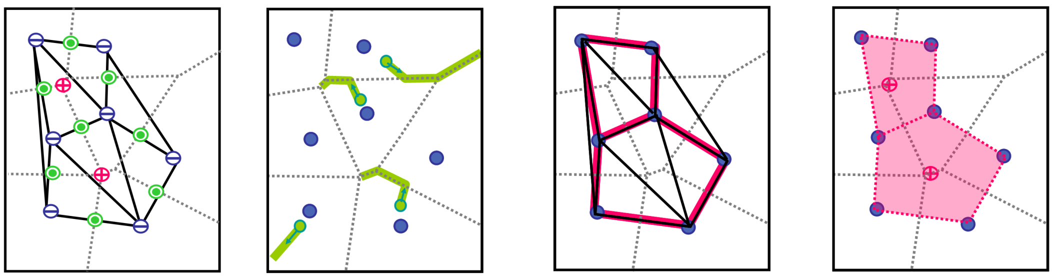


Figure 2.7: Flow complex. From left to right: 1. The local minima $\ominus$, saddle points $\odot$ and local maxima $\oplus$ of the distance function induced by the sample points (local minima). 2. Some orbits of the flow induced by the sample points (blue). 3. The stable manifolds of the saddle points. 4. The stable manifolds of the local maxima.

The main idea is to study where a point in $\mathbb{R}^3$ flows when following the direction of steepest ascent of the distance function to the sample points. It turns out that all points flow into a local maximum (fig.2.7). The set of all points that flow into a critical point is called the stable manifold of this critical point. the collection of all stable manifolds is called the **flow complex** of the sample points. The reconstructed surface is the union of the stable manifolds of the maxima.

Algorithm: The flow complex is not a subcomplex of the Delaunay $\mathbf{D}(P)$ triangulation though $\mathbf{D}(P)$ can be used to compute the flow complex. This computation is quite involved and makes use of the recursive structure of the stable manifolds.

2.1.6 EigenCrust

The *EigenCrust* was proposed by Kolluri et al. [KSO04] in order to produce watertight surface.

In this approach, the Delaunay tetrahedralization of the sample points is computed, then a variant of spectral graph partitioning is performed to decide whether each tetrahedron is inside or outside the original object, i.e whether a pole is inside or outside. At the end, the reconstructed surface triangulation is the set of triangular faces where inside and outside tetrahedra meet.

The Eigencrust algorithm handles undersampling, noise, and outliers by using a spectral graph partitioner to obtain a robust tetrahedra classification.

A pole graph (fig.2.8) is constructed. The nodes represent the poles. There is an edge $e_{(i,j)}$ between two poles v_i and v_j if:

1. the poles are neighbors in the Delaunay triangulation of the poles, or

2. the two poles share a vertex.

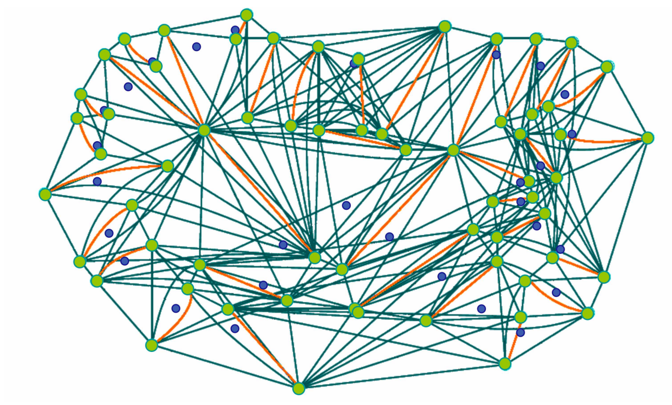


Figure 2.8: Pole graph. The negative weighted egdes (orange) and the positives ones (dark green). The weight is large if the maximal balls intersect deeply. (The input point (blue) and the poles (green))

The edges, $e_{(i,j)}$, of the graph are weighted according to the likelihood that the two poles lie on the same side of the surface. In case 1, the weight is positive. In the case 2, the weight is negative (respectively the green and the orange edges in the figure 2.8). The weighted graph is represented by the pole matrix, $L(i,j) = -w(i,j)$. The eigen vector associated to the smallest eigen value of the pole matrix L determines a division of the graph into two subgraphes containing inside and outside poles.

In addition to be Delaunay based, the Eigencrust can be seen as a graph cut algorithm (sec.2.3.1). The *Normalized Cut* is performed, the pole graph is partitioned using two criteria: minimizing sum of weights of the cut edges and cutting the graph into two pieces of equal size.

2.2 Functionnal Based Surface Reconstruction

The surface S may be defined by functions implicitly (see sec.1.1). The object surface, S, is characterized by

$$S = \left\{ x \in \mathbb{R}^3 : f(x) = 0 \right\}, \tag{2.2}$$

where f is the signed distance from x to the surface object S.

S is unknown, the distance function can not be computed exactly, thus an approximation must be performed. The problem is now to define the (signed) distance function f.

Usually the signed distance function is approximated as a composition (weighted sum) of simple primitives f_i (such as blobs, quadric, radial basis function,...) to find a scalar function such that all data points are close to an isocontour of the scalar function. The distance function may be defined globally or locally.

For some applications, computing the function f all over the space is not necessary and partitioning the space into inside or outside area is sufficient. In this case, the method try to reconstruct an indicator function instead of distance function (sec.2.2.6).

Functional based approaches are robust when the data points are unorganized and non uniform. However their computation costs are often high for large data sets since the construction is global which results in solving a large linear system.

2.2.1 Tangent Planes Estimation

Hoppe et al. [HDD$^+$92] proposed a signed distance function f based on an estimation of the oriented tangent planes. For each data point p_i, a tangent plane is computed by least-squares approximation based on a principal component analysis (PCA) on the k nearest neighbors of p_i (Fig.2.9). A consistent orientation is obtained by solving a graph optimization problem with a minimum spanning tree (MST). The signed distance function at a point $q \in \mathbb{R}^3$ is approximated by the distance between q and the nearest tangent plane. As the sampling ideally should be proportional to the curvature in some sense,

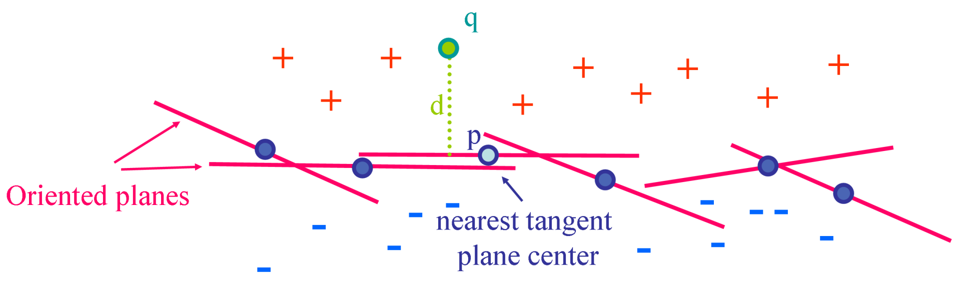


Figure 2.9: Signed Distance function. For a given point q, its nearest neighbor p in the set of input point (blue) is found then the distance d to the estimated tangent plane in p is computed.

Curless Levoy [?] propose a modification which is one of the best results in practice.

These algorithms require a uniform sampling at least locally since otherwise the k nearest neighbors may be almost collinear, resulting in a poor estimation of the tangent plane.

In [?], Boissonnat an Cazals proposed a smooth surface reconstruction via natural neighbor interpolation of distance functions. This method combines Voronoi diagrams and implicit functions. It works in any dimension and is suitable for surfaces of arbitrary topology and non-uniform sampling. Given a Delaunay triangulation, the neighborhood of a vertex is naturaly defined as the set of vertices connected to that vertex. This information

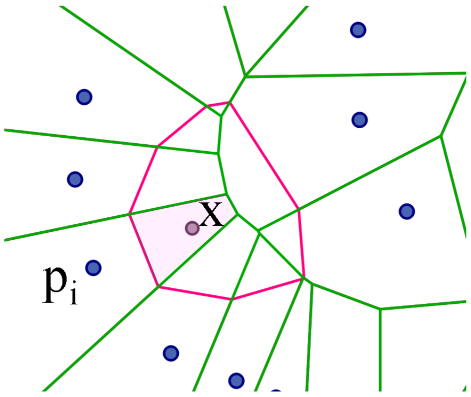


Figure 2.10: Natural Neighbors. Point x has 8 natural neighbors. The red cell is created when the point x is added to the Voronoi diagram (green). The normalized surface of the pink area correspond to the coordinate of x according to p_i

is of combinatorial nature and can be made quantitative using the so-called *natural coordinates*. The natural coordinate of a point x according to a point $p-i$ of P is the normalized measure of the region of withdrawn from p_i if x is added in the Voronoi diagram (fig.2.10)

2.2.2 Skeleton Based Implicit Model

Blinn [Bli82] developed the first skeleton-based implicit model which is now called *Blobby Model*. A skeleton is composed of a collection of geometrical primitives such as points or lines. which can be represented by a tree or a graph. Each primitives is associated to a potential function, $v(x)$, which decreases according to the distance from the primitive to x.

In the blobby model approach, the skeleton is formed by a set of points $P = \{p_i\}$. The potential functions are defined by a class of Gaussian function D_i centered at each point p_i of the skeleton (2.3).

$$D_i(x) = b_i * exp(-a_i \cdot r_i(x)^2) \tag{2.3}$$

where a_i and b_i are scalar value, respectively the spread and the mean of D_i, and $r_i(x) = \|x - p_i\|$. In the case of a spherically symmetric field, $r_i(x)$ is the euclidean distance from x to the skeleton point p_i.

Muraki et al [Mur91] proposed a reconstruction algorithm based on the *blobby model*. The surface S' is defined implicitly as the zero-level set of a field function f expressed as a linear combination of three-dimensional Gaussian kernels with different means and spreads (called *Blobby Primitives*) (2.4).

$$f(x) = \sum_{i=1}^{N} D_i(x) \tag{2.4}$$

Muraki's approach is to make an initial fit between a blob and the data, and to divide the blob into two blobs so as to increase the goodness of the fit by solving an energy minimization problem (2.5) (Fig.2.11):

$$E = \sum_{i=1}^{n} (f(p_i) - iso)^2 \tag{2.5}$$

where iso is a field value and f a field function. This algorithm needs a partitioning of the

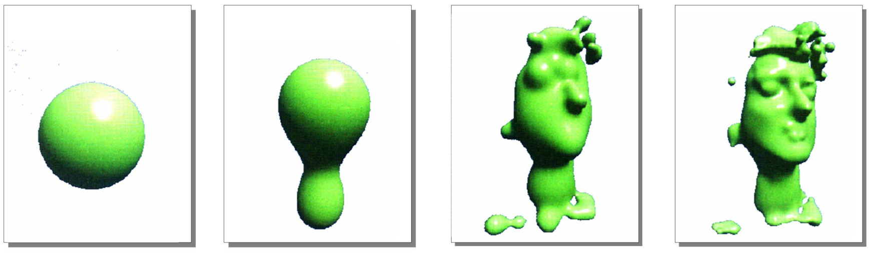


Figure 2.11: Blobby Model. The transformation of a "Blobby Model" with the number of primitives N. From left to right : $N = 1, 2, 35, 243$

space into inside and outside as the algorithm might be modeling the outside instead of the inside.

Although the output is always watertight, bubbly-shaped, the convergence rate is very slow and the algorithm is computationally expensive. To handle this problem, Tsingos et al [?] reduce the computation time by using the medial axis which brings more relevant information to chose the skeleton points, i.e. the primitives.

In their algorithm, the medial axis is defined as the set of points that are the farthest inside the object. In order to extract the medial axis, a (Chamfer) distance map is computed. This map is based upon a voxel approximation of the object using a 3D grid and a voxel labeling as inside or outside.

The points are selected in the medial axis point with a greedy algorithm: candidate points are added where the reconstruction is not accurate enough. In this approach, the field function f is radial and compactly supported.

2.2.3 Surface Fitting

Another approach is to perform some type of surface fitting with a polynomial [?] or an algebraic surface [?] to the data.

The main idea is to choose a model such as polynomial (quadric, B-splines,...) and to seek for the best parameters of the model in order to fit the data points. More precisely, the function/model f is fitted to the data P by minimizing the squared distances between the points p_i and the model f, i.e. by minimizing a least squares error (2.6) :

$$f^* = argmin_f(\sum_{i=1}^{n} (f_i - f(p_i))^2) \tag{2.6}$$

Several approaches add a regularization term to the error in order to smooth the result. The error to minimize is defined as:

$$E(f) = \sum_{i=1}^{n} (f_i - f(p_i))^2 + \lambda D(\|f\|) \tag{2.7}$$

where λ is the regularization parameter which allows to tune the smoothness of the result and D is a differential operator which brings an a priori on the kind of desired smoothness. These approaches are global, that is, for example the quantity of smoothness is the same for all shapes.

Othake et al [OBT+03] proposed a quadric fitting which preserve sharp features by using a multi level partition of unity and three kinds of fitting model. The surface is defined locally with quadric functions that are blended together globally by partition of unity.

Partition of Unity *"Divide and conquer"* is the main idea behind the *Partition of Unity* approach. The main idea consists of breaking the domain Ω into M smaller mildly overlapping subdomains $\{\Omega_i\}_{i=1..M}$ where the problem can be solved locally. On each subdomain Ω_i, the data are first approximated, and the local solutions f_i are blended together using a weighting sum of local subdomain approximations (2.8).

$$f(x) = \sum_{i=1}^{M} w_i(x) f_i(x). \tag{2.8}$$

The weights w_i are smooth functions and sum up to one everywhere on the domain. Then determine the continuity of the global reconstruction function f. The condition $\sum_{i=1}^{M} w_i = 1$ can be obtained from any set of smooth functions W_i by a normalization process:

$$w_i(x) = \frac{W_i(x)}{\sum_{j=1}^{n} W_j(x)}. \tag{2.9}$$

In [OBT+03], the fitting strategy is adapted to each cell of the octree.According to the number of points and distribution of associated normals, different functions are used to perform a local fitting.

The local fitting strategy depends on the number of points in a given cell and on the distribution of normals of those points. At a given cell the most appropriate of these three local approximations is used: a general 3D quadric fitting, a bivariate quadratic polynomial fitting in local coordinates, or a piecewise quadric surface that fits an edge or a corner. In order to detect sharp features, a clustering of the normal is performed (Fig.2.12).

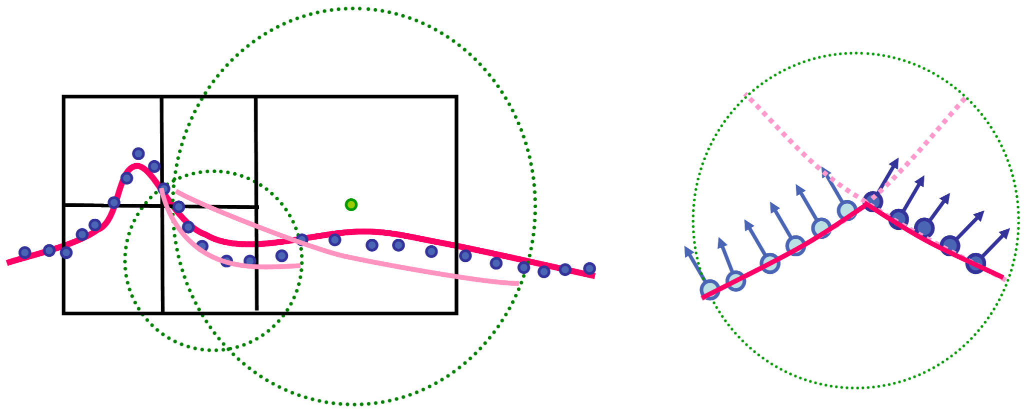


Figure 2.12: Multilevel partition of unity implicit. Left: an octree is computed according to the distribution of the data points (blue). For each cell of the octree, a quadric surface (light pink) is fitted to the points contained in a ball centered at the cell. The type of quadrics fitted is chosen according to the distribution of the points. Then, the local reconstructions are blended to obtain the surface (dark pink). Right: illustration of the case where an edge is detected by normal clustering.

2.2.4 Moving Least Squares

The idea of the Moving Least Square (MLS) approach [PK81] comes from the least squares technique to fit a surface to a set of points described section 2.2.3.

Given a point cloud P, the surface S' is reconstructed by applying a projection operator ψ that moves the points in the vicinity of P onto the inferred surface S. The surface S' is thus defined as the set of fixed points of the projection operator.

Definition 2.4. *The Moving Least Square surface for a point cloud $P \subseteq \mathbb{R}^3$ is defined as the set of stationary points of a certain projector function $\psi : \mathbb{R}^3 \to \mathbb{R}^3$:*

$$S = \{x \in \mathbf{B} : \psi(x) = x\}\,, \tag{2.10}$$

where $\mathbf{B}$ is a tubular neighborhood of the MLS surface.

The tubular neighborhood (fig.2.13:Left), $\mathbf{B}$, can be defined as an union of balls centered at the points p_i.

$$\mathbf{B} = \left\{x \in \mathbb{R}^3 : \|x - p_i\| < r_{\mathbf{B}}\right\}. \tag{2.11}$$

Let x be a point in a neighborhood of P. A local reference plane H_x for x is determined by minimizing a local weighted sum of squared distance of the points p_i to the plane H_x:

$$E_{\vec{n},q}(x) = \sum_{i=1}^{N} (\vec{n} \cdot (p_i - q))^2 \; \theta(|p_i - q|), \tag{2.12}$$

where $\vec{n}$ is the unit normal to H_x, $q = x + d\vec{n}$ the projected point of x on H_x, d is

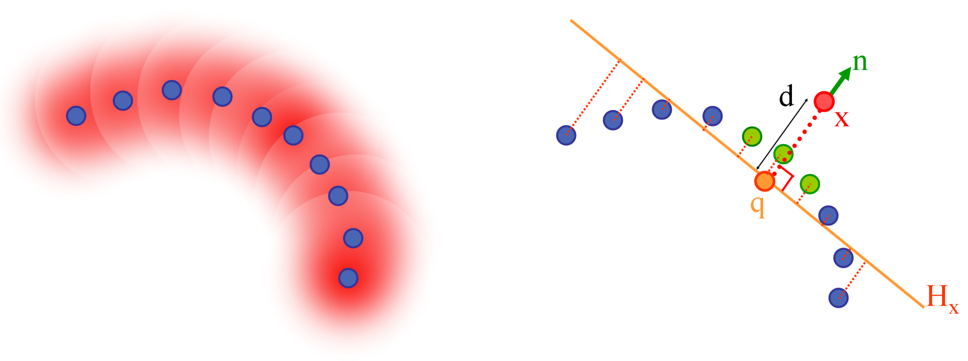


Figure 2.13: Moving least squares. Left: the tubular neighborhood **B** of the blue point set. Right: A local reference plane H_x for the points x (red) is determined by minimizing a local weighted sum of squared distances of the points p_i (blue and green) to the plane H_x. q (orange) is the projection of x onto H_x.

the distance from x to the plane and θ is a radially symmetric, positive and monotically decreasing weight function. The algorithm minimizes iteratively the energy function $E_{\vec{n},q}$ in order to find $\vec{n}$ and q to determine the projector Φ.

Several definitions have been proposed for defined the projector function ψ and θ. The projection algorithm is also derived in several variants.

Alexa et al. [ABCO+01] proposed to add a polynomial fitting step which can be applied after the map ψ. The local reference plane H_x is used to compute a local bivariate polynomial approximation $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ of the surface in a neighborhood of x, using local coordinates. The weighted energy is defined by:

$$E = \sum_{i=1}^{N} (f(x_i, y_i) - d_i)^2 \; \theta(|p_i - q|), \tag{2.13}$$

where (x_i, y_i) are the coordinates of the input points p_i in the local reference plane H_x and d_i is the distance from p_i to the plane. Alexa uses cubic or quartic polynomials for f , since they experienced surface oscillation artifacts with higher degree polynomials.

In the Adaptive MLS (AMLS) [?], the weight θ is adapted to the local feature size (lfs) in order to adapt the weight to the local geometry of the point cloud.

In [?], Reuter et al. presented an alternative projection operator based upon the Enriched Reproducing Kernel Particle Approximation (ERKPA), that allows not only to reproduce polynomials, but also some richer functions with discontinuous derivatives. A user specifies the locations of the discontinuities that generate the sharp features in the resulting point set surface. A compactly supported enrichment function with a user-specifiable support size allows controlling the influence domain of the sharp feature (fig.2.14).

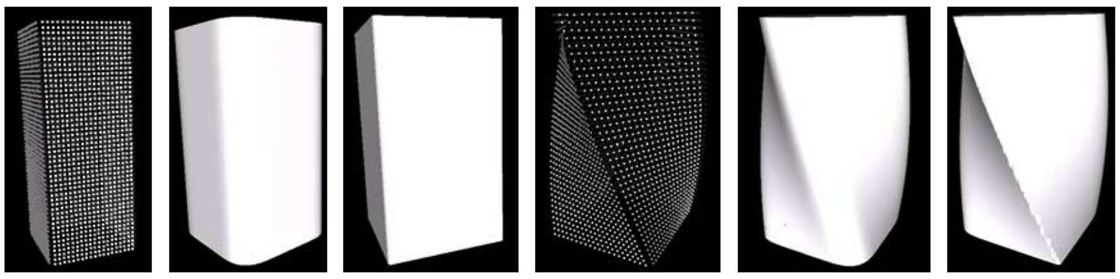

(a) Original cloud (3900 points)
(b) MLS
(c) ERKPA
(d) Twisted cloud (3900 points)
(e) MLS
(f) ERKPA

Figure 2.14: Surface reconstruction with sharp edges by MLS vs. ERKPA

2.2.5 Radial Basis Functions

About 20 years ago, in an extensive survey, Franke [FN80] identified Radial Basis Functions (RBF) as an relevant method to solve scattered data interpolation problems. RBFs are used to reconstruct smooth, manifold surfaces from point-cloud data and to repair incomplete meshes.

The surface is defined as the zero level set of a function f defined from a class of basis functions $\Phi : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}$, as a linear combination

$$f(x) = \sum_{j=1}^{m} \alpha_j \Phi(x, c_j), \qquad (2.14)$$

where $\{c_j\}_{j=1...m}$ denotes a set of m points and $\{\alpha_j\}_{j=1...m}$ denotes a set of unknown weights to be solved for.

As the invariance according to the rotation and the translation is required, the function Φ defined as a radial function $\Phi(x, c) = \phi(\|x - c\|)$ where c is a center and $\phi : \mathbb{R} \rightarrow \mathbb{R}$ can be linear, biharmonic or polyharmonic.

Reconstruction using RBFs provides us with smooth implicit interpolating surface, since, both the implicit solution and its zero level set share the same continuity properties as the ones of the basis function Φ.

Many RBF-based surface reconstruction algorithm have been proposed [CFB97, DTS00, OBS03, TI04, CBC$^+$01, Sch95, BL97, Buh03, ?, OBS03, OBS04, Wen02, Wen95, Wu95]. As our approach is based on RBF, we will review the corresponding state of the art in chapter 3.

2.2.6 Indicator Function

In [?], Kazhdan developed a surface reconstruction technique in which solve for an indicator function (2.15) of the shape (fig.2.15). Let $\mathcal{O}$ be the object such that S is its boundary.

$$\chi_{\mathcal{O}}(p) = \begin{cases} 0 & \text{if p} \notin \mathcal{O} \\ 1 & \text{if p} \in \mathcal{O} \end{cases} \qquad (2.15)$$

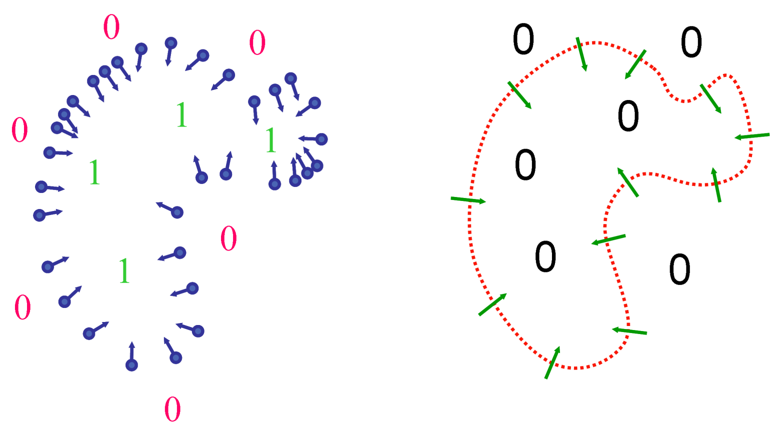


Figure 2.15: Poisson surface reconstruction. Left: the input point with theirs normals. The value of the indicator function $\chi_{\mathcal{O}}$ are 0 outside and 1 inside the object. Right: the indicator gradient is $\nabla\chi_{\mathcal{O}}$ non zero on a neighborhood of the input points.

Algorithm: Given a set of input points P and their oriented normals, the algorithm constructs an octree which is used to estimate a vector field of the normals. The indicator function f is then computed by solving a Poisson equation, where the normal field $\vec{n}$ is specified as the gradient of the indicator function:

$$\nabla f(x) = \vec{n}(x) \;\; \forall x \in \mathbb{R}^3. \tag{2.16}$$

By using the divergence operator, the equation (2.16) became:

$$\Delta f(x) = (div)\vec{n}(x) \;\; \forall x \in \mathbb{R}^3. \tag{2.17}$$

The divergence of the vector field is computed on the cells of the octree and the equation (2.17) is solved by a *Fast Fourier Transform* (FFT) and a multiscale linear solver [**?**].

In [**?**], Schall et all proposed a variant of the Poisson reconstruction called *adaptative Fourier-based* surface reconstruction. This approach is based on the partition of unity and they perform an error guided subdivision of the input data points. The decomposition of the space is based on an octree. A local solution is computed as a local characteristic function for the points inside the cell using Kazhdan global approach. But since the points inside a cell do not represent a solid, the characteristic function may not be really computed using the Poisson global approach. To avoid surface parts created which are not represented by points, the solution is to subdivide the octree cells if the resulting local approximation inside the cell is not accurate enough. By iterating this procedure, overlapping local characteristic functions are computed for each octree leaves for each part of the input with a user-defined accuracy. The final reconstruction is then obtained by combining the local approximations using the partition of unity approach and extracting the surface using a polygonization algorithm.

This algorithm is faster than the one of kazhdan but the characteristic function is only determined closed to the surface and not for the whole volume.

2.3 Others

2.3.1 Graph Cut Based Reconstruction

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

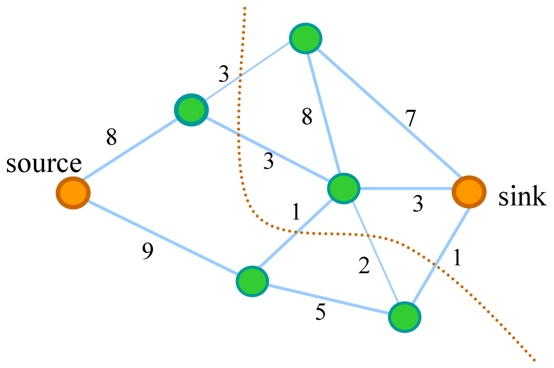


Figure 2.16: Min cut in 2D. Given a graph with valued edges (e, w_e), find min cut between source and sink.

An undirected graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is defined as a set of nodes (vertices $\mathcal{V}$) and a set of undirected edges ($\mathcal{E}$) that connect these nodes. An example of such graph is shown in Figure 2.16. Each edge $e \in \mathcal{E}$ in the graph is assigned a nonnegative weight (cost) w_e. There are also two special nodes called *sink* and *source*. A cut is a subset of edges $C \subset \mathcal{E}$ such that the sink and the source become separated on the induced graph $\mathcal{G}(C) = \langle \mathcal{V}, \mathcal{E} \setminus C \rangle$. Each cut has a cost which is defined as the sum of the costs of the edges that it severs:

$$cost = \sum_{e \in C} w_e. \tag{2.18}$$

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

Hornung and Kobbelt [AH06] present a volumetric method for reconstructing watertight triangle meshes from arbitrary point clouds (Fig.2.17) in which normal orientation is not required. Their approach uses an unsigned distance function, hence no requirement about the local surface orientation.

The figure 2.17 describes the algorithm. First, the input points, P, are inserted into a volumetric grid. This leads to a grid with a sparse set of occupied voxels. A confidence map is computed: at each voxel v is associated a probability $\phi(v)$ that the surface intersects v. A crust containing the surface is computed using morphological dilation and a medial axis

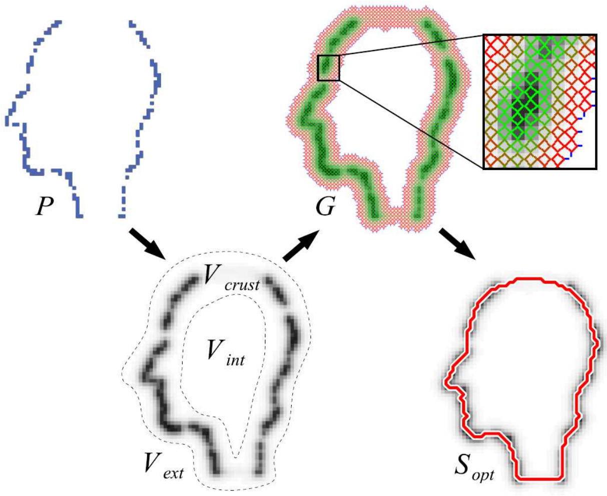


Figure 2.17: Reconstruction process in 2D

approximation. The unknown surface is supposed to lie in the voxel crust V_{crust} between the outer boundary V_{ext} and the inner boundary V_{int}. A unsigned distance function is computed by volumetric diffusion. A spatial graph structure G is embedded within the voxel grid. In G, a graph node is associated with each voxel face, and a weighted graph edge is created for each voxel edge, such that each voxel contains an octahedral subgraph. G is weighted such that small edge weights for high confidence voxels and vice versa. The boundaries are connected a sink and a source node, respectively. Computing the min-cut of this graph yields the surface S_{opt} .

2.3.2 Statistical Based Reconstruction

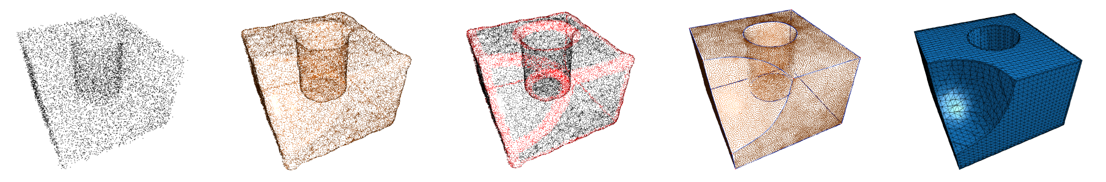
Statistical approaches were proposed to deal with noise in the input points and to reconstruct a piecewise surface.

Assuming a given real-world scene S and a measurement D, the data, i.e. the input points. Then the probability space of $\Omega = \Omega_S \times \Omega_D$, the set of all possible real-world scenes and of all possible measurements of them, is considered. The measurement D is created from S by a process involving statistical errors. Note that assuming an unbiased measurement process, the most probable original scene is still the measurement itself.

The Bayesian statistics approach to this problem is defining a probability distribution $P(S)$ over the set of all possible original scenes. Then, we can apply Bayes rule to invert the measurement process in a statistical sense. The probability of a reconstruction S being the original scene given measurement D is computed as:

$$P(S|D) = \frac{P(D|S)P(S)}{P(D)} \tag{2.19}$$

where $P(D|S)$ is the probability distribution of the likelihood of measurements D being made of scenes S and $P(S)$ is the probability distribution. Note that $P(S)$ is usually not



(a) Noisy input point cloud. (b) Initial smoothing. (c) Estimating edge probabilities. (d) Smoothing with discrete attributes. (e) Triangulation of the reconstruction.

Figure 2.18: Bayesian point cloud reconstruction

be an exact probabilistic model of all potentially measured scenes but only a description of partial prior knowledge or belief of reasonable models. Prior probabilities are the key to any Bayesian reconstruction technique. They define what artifacts are considered noise and thus what the reconstructed scene will look like.

In order to find the most likely reconstruction, the scene S is determined by maximizing $P(S|D)$, called maximum a posteriori solution SMAP. As the denominator in Equation (2.19) is only a normalization constant, not depending on S, it is sufficient to compute:

$$S_{MAP} = \arg\max_S P(S|D) = \arg\max_S PS|D)P(S) \tag{2.20}$$

The challenge is to define probabilistic model of the measurement process and of the prior.

In the Bayesian point Cloud Reconstruction [?], let an original scene be a set of n points chosen according to the probability density $P(S)$. A measurement process deletes some of the original points, leaving $m \leq n$ measured points D. Then, random noise is added to the remaining points according to a density $p(D|S)$ (Fig.2.18(b)).

To invert the measurement process in a statistical sense, an estimate point cloud S with size $n \geq m$ is constructed.

After this initialization, the algorithm seek for a reconstructed surface S' that maximize the posterior probability $p(S'|D)$ by numerical optimization.

The prior model include both assumptions on continuous properties as well as discrete properties for representing sharp features. The objects are assumed to consists of piecewise smooth patches separated by sharp boundaries. The priors consists of three main ingredients: density priors, smoothness priors and priors for estimating sharp features (2.21).

$$P(S) = \frac{1}{Z} P_{\text{density}}(S) P_{\text{smooth}}(S) P_{\text{descrete}}(S) w(S) \tag{2.21}$$

where Z is a normalization constant and $w(S)$ is a box function such that $w(x) = 1$ if x is inside the box and 0 outside .

2.3.3 Deformable models

Deformable models have been first introduced by Kass et al. in late 80s [KWT88], for the purpose of image segmentation. It has been proved successful in surface reconstruction from point cloud for medical applications.

The approaches based on deformable models combine knowledge from mathematics, physics and mechanics. The surface is obtained as the final sate of the evolution of a given surface. A deformable model is a geometric object whose shape can change over time (fig.2.19). The deformation behavior of a deformable model is governed by variational principles (VPs) and/or partial differential equations (PDEs).

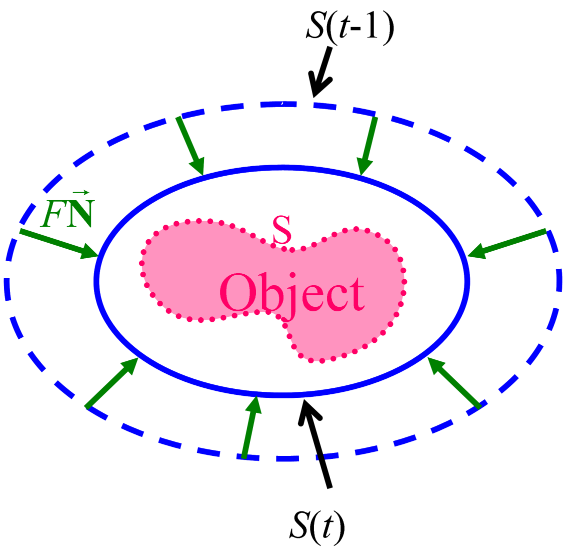


Figure 2.19: Active contours in 2D. The curve S is approximated dynamically by a family of curves $S(t), t = 0 \ldots n$. The curve $S(t)$ evolves to $S(t+1)$ according to $F\vec{N}$

The deformation algorithm is based on three components: the geometric representation (explicit, implicit or parametric), the evolution law and the topology control. As the evolution law is known, the topology may be constrained to not change.

The evolution law is governed by a partial differential equation. The approach considers a family of surfaces $S(p,t)$ in $\mathbb{R}^3$, where p parameterizes the surface and t is the time, that evolve according to the following PDE (2.22):

$$\frac{\partial S}{\partial t} = F(S, N, K, f, ...), \tag{2.22}$$

where F is an evolution functional, N is the surface normal, K is the surface curvature and f is a function of the internal or/and external force. The initial condition is $S(t=0) = S_0$ and S_0 is some initial closed surface.

Generally, the deformation involves a data term and a regularization term. The data term drives the model deformation toward the boundary and the regularization term enforces a smooth behavior of the model.

Algorithm based on deformable models have been used to reconstruct surfaces from point sets.

There is a lot of algorithms based on Deformable Models. Several of them perform a surface reconstruction from point set. For example, Zhao et al. [ZOF01] proposed a weighted minimal surface model based on the potential functional related to the distance to the points. Notes that this approach, called *level set* was originally introduced by Osher and Sethian [OS88], where an initial surface is continuously deformed in order to fit the data points. The evolution is guided by the gradient of the functional F until reaching an equilibrium.

At each step, every point x of the surface $S(t)$ evolves toward the interior of the surface, along the normal direction to $S(t)$ at point x, with a displacement speed that is proportional to:

$$-\nabla d(x) \cdot \vec{n} + t(x) \tag{2.23}$$

K denotes the mean curvature of the surface at x and $\vec{n}$ is the inner normal at x.The tension of the surface represented by the second term $t(x)$ is not linear, so that the evolution process requires a huge number of steps before reaching its equilibrium.

Chaine [?], in the *geometric convection* algorithm, incrementally sculpts the Delaunay triangulation of the input point. The sculpting strategy is a discrete algorithm equivalent to the evolution equation of Zhao. This algorithm is based on the 3D Delaunay triangulation and on an oriented watertight pseudo-surface maintained during evolution.

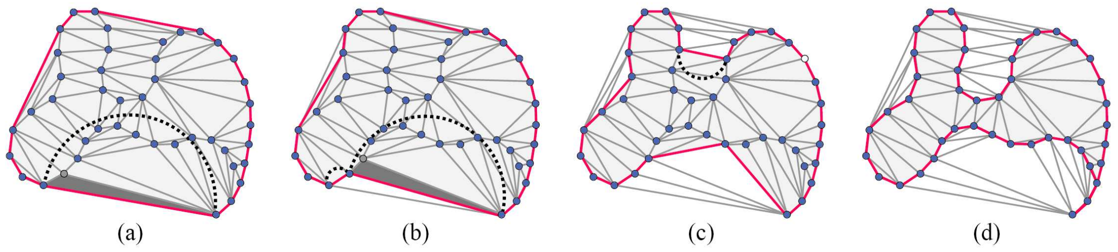


Figure 2.20: Geometric convection in 2D. From left to right: (a) Initialization of the curve by the convex hull of the point set. (b) The curve is updated. (c) An edge block a cavity. (d) the resulting curve.

At the beginning the pseudosurface is initialized as the convex hull of the point set (Fig.2.20(a)) and then the pseudosurface is deformed using a set of geometric and topological operations (Fig.2.20(b)). The convection process corresponds to the first term of the evolution equation (2.23) proposed by Zhao. A possible extension of the algorithm is proposed in which the second term of the equation (2.23) is not explicitly rejected. This approach allow to resolves undetected pockets (Fig.2.20(c)).

In [?], Sharf et al present a deformable model that uses an explicit evolving front technique for reconstruction of a 3D model. Their model includes multiple competing evolving fronts at different locations that approach towards the finer local features of the target shape only after reconstructing the coarse global features (fig.2.21). The front evolution is guided by a scalar-field representing the distance from the point set (in outward normal direction). Thus, the front evolution is adapted to the local feature size of the shape.

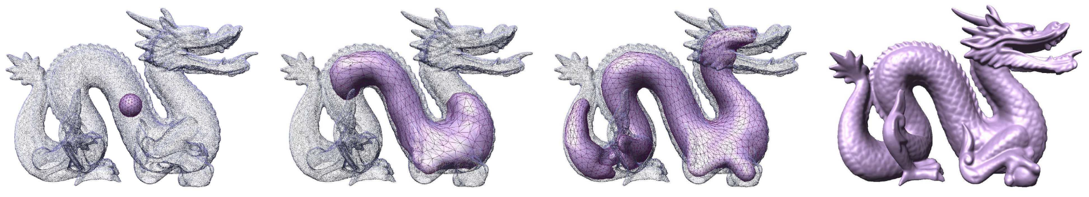


Figure 2.21: Reconstruction with a deformable model. From left to right (3 first images): Growing of a watertight (genus 0) mesh model inside the point cloud of the dragon. Right: the handle between the body and the tail is attached and the model is projected onto the point cloud.

Chapter 3

Radial Basis Functions

Radial Basis Functions (RBF) were first introduced by Broomhead and Lowe in the neural network literature [BL88]. Then, the RBF techniques became standard tools for pattern recognition [Kir01], statistical learning [HTF01] and, about 20 years ago, in an extensive survey, Franke [FN80] identified radial basis functions as accurate tools to solve scattered data interpolation/approximation problems.

Among the many techniques developed for surface reconstruction with implicit methods, the RBF approach has shown successful at reconstructing surfaces from point sets scattered on surfaces S of arbitrary topology. Furthermore, the reconstructed surface S' is watertight, holes are filled.
Remarks: Carr et al. in [CFB97] proposed to fit skull surface using RBF algorithm in order to repairing defect, usualy hole in the skull with cranial implant (cranioplasty).

The RBF methods can be seen as variational problems [Duc77] which mathematical properties have been widely explored [Buh03, Isk04, Wen04].

3.1 Least-Squares Approximation

Definition 3.1. *The* ***least-squares approximation problem*** *is formulated as follows. Given $X = \{x_i\}_{i=1\ldots n}$ a set of n points in $\mathbb{R}^3$ and n scalar numbers $F = \{f_i\}_{i=1\ldots n}$ find a function in $\mathcal{F} = \{f : \mathbb{R}^3 \to \mathbb{R}\}$ satisfying the approximation condition:*

$$f^* = \arg\min_{f\in\mathcal{F}} E(f), \tag{3.1}$$

where E is the energy functional given by the least-squares error :

$$E(f) = \sum_{i=1}^{n} \left(f_i - f(x_i)\right)^2 . \tag{3.2}$$

A common approach is to reduce the space of functions where the optimal function is searched to a finite subset, generated by a finite set of basis functions $\Phi_j : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$, as a linear combination:

$$f(x) = \sum_{j=1}^{m} \alpha_j \Phi_j(x), \tag{3.3}$$

where $\{\alpha_j\}_{j=1\ldots m}$ denotes a set of unknown weights to be determined.

Definition 3.2. *We called* ***constraints*** *the set of points, $X = \{x_i\}$, where the function value, f_i, is specified.*

Therefore, minimizing the energy (3.2) consists in solving a linear system with least-squares technique. That is, to determine the vector $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ by solving a linear system

$$G^t_{X,\Phi} G_{X,\Phi} \alpha = F \tag{3.4}$$

where

$$G_{X,\Phi} \begin{pmatrix} \Phi_1(x_1) & \Phi_2(x_1) & \ldots & \Phi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_1(x_n) & \Phi_2(x_n) & \ldots & \Phi_n(x_n) \end{pmatrix} \tag{3.5}$$

3.1.1 Regularization Theory

When the reconstruction problem is ill-posed, a solution to solve it is to add a prior term in the energy functional. The main idea underlying regularization theory is a solution can be obtained from a variational principle which contains both the data and the prior regularity information. The prior wishes on the function regularity can be a surface with a minimum curvature or with a minimum area.

That is, a regularization term $\mathcal{E}_r(f)$ is added to the least square term, $\mathcal{E}_s(f)$, in the energy functional E (3.2). Thus, the energy functional is defined as (3.6)

$$E_\lambda(f) = \mathcal{E}_s(f) + \lambda \mathcal{E}_r(f) \tag{3.6}$$

where $\mathcal{E}_s(f) = \sum_{i=1}^{n} (f(x_i) - f_i)^2$ is the least square error, $\mathcal{E}_r$ is the regularity prior term and λ is positive scalar value, called *regularization parameter*, which allows to tune the degree of smoothness. When $\lambda \to 0$, the surface closely fits the constraints. Conversely for largest values of λ, the surface approximates the data points by increasing the regularity of the function (fig.3.1).

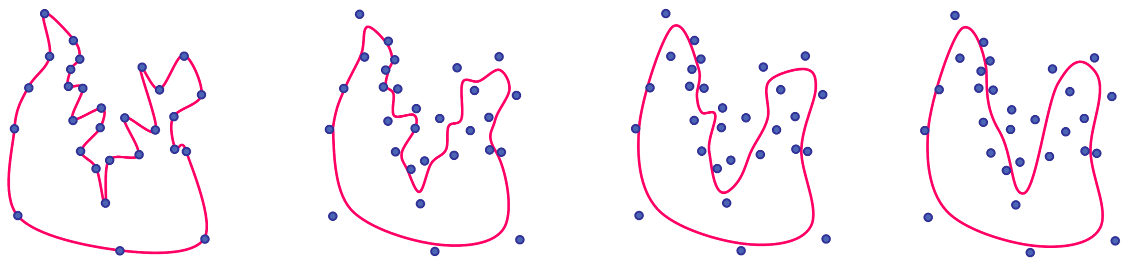


Figure 3.1: Regularization theory. From left to right the λ value increase.

In most cases, the regularization term is defined using a linear differential operator, $\mathbb{D}$:

$$\mathcal{E}_r = \|\mathbb{D}f\|^2 \tag{3.7}$$

The problem to be solved is the minimization problem (3.1) with an energy functional E_λ which is λ-dependent (3.8), λ is fixed.

$$\mathcal{E}_\lambda = \sum_{i=1}^{n} (f_i - f(x_i))^2 + \lambda\|\mathbb{D}f\|^2. \tag{3.8}$$

3.1.2 Interpolation Problem

In the case of interpolation problem, the function f must verify all the constraints, i.e. the least square error must be zero thus f satisfies $E(f) = 0$.

Definition 3.3. *Interpolation problem :*
Given $X = \{x_i\}_{i=1\dots n}$ a set of n points and n scalar numbers $F = \{f_i\}_{i=1\dots n}$, find a function $f : \mathbb{R}^3 \to \mathbb{R}$ satisfying the interpolation constraints :

$$f(x_i) = f_i \qquad \forall i = 1\dots n. \tag{3.9}$$

So by definition, the function f is such that

$$f_i = \sum_{j=1}^{m} \alpha_j \Phi_j(x_i) \ \ \forall x_i \in X \tag{3.10}$$

The reconstruction problem thus consists in determining the vector $\alpha = [\alpha_1, \dots, \alpha_n]$ by solving a linear system of equations given by the *constraints* (3.9):

$$G_{X,\Phi} \cdot \ \alpha = F. \tag{3.11}$$

where $G_{X,\Phi}$ and F are the same than in (5.9).

The problem has a solution if $m \geq n$, the solution is unique if $m = n$.

3.2 Radial Basis Functions

In the RBF approach, the basis functions are centered on some particular points called *centers*. Let $C = \{c_i\}_{i=1\dots m}$ be a set of centers. The functions Φ in (3.3) are then of the form:

$$\Phi_j(x) = \Phi(x, c_j). \tag{3.12}$$

For geometric application, we require the solution to be invariant by rigid transformation. We constrain the solution to be stable to translation and rotation of the point set.

The function Φ is thus restricted to the set of **radial** functions :

$$\Phi(x, c_j) = \phi(\|x - c_j\|) \tag{3.13}$$

where $\phi : \mathbb{R}^+ \rightarrow \mathbb{R}$ and $\|.\|$ denote the Euclidean distance.

All the RBF theory was developed in the interpolation problem case (sec.3.1.2) and the centers of the RBF are chosen to coincide with the constraints (3.14).

$$c_i = x_i \ \forall i = 1 \dots n \tag{3.14}$$

In the following, the functions Φ_i in (3.13) are then of the form:

$$\Phi_i(x) = \Phi(\|x - x_i\|) \ \forall i = 1 \dots n. \tag{3.15}$$

This choice allows the system (3.11) to be square. That is the matrix $G_{X,\Phi}$ is square:

$$G_{X,\Phi} = \begin{pmatrix} \phi(\|x_1 - x_1\|) & \phi(\|x_1 - x_2\|) & \dots & \phi(\|x_1 - x_n\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|x_n - x_1\|) & \phi(\|x_n - x_2\|) & \dots & \phi(\|x_n - x_n\|) \end{pmatrix} \tag{3.16}$$

The main question is: how can we be sure that the interpolation matrix $G_{X,\Phi}$ is non singular ?
Micchelli [Mic86] has shown the distance matrix generated by distinct points to be invertible for several useful choices of ϕ. A good overview was done by Buhmann [Buh03] , Dyn [?] and Powell [?]. One of the most attractive features of radial basis function methods is the fact that, for most choice of basis functions ϕ, a unique interpolant is guaranteed under rather mild conditions on the centers but this is not always the case. For example, one important exception to this statement is the *Thin plate spline* introduced by Duchon in order to produce an approximation which minimize the curvature in 2D. The thin plate spline, in 2D, is the solution to a variational problem: the minimization of the integral (3.17):

$$\int_{\mathbb{R}^2} \left(\frac{\partial f}{\partial x_1 \partial x_1}\right)^2 + 2\left(\frac{\partial f}{\partial x_1 \partial x_2}\right)^2 + \left(\frac{\partial f}{\partial x_2 \partial x_2}\right)^2 \tag{3.17}$$

Remarks: this integral can be seen as a norm of a differential operator, i.e it can be formulated as $\|D^2(f)\|$. Thus, it is possible to link this problem to the regularization theory (see sec.3.1.1) [Hay99].

Duchon has shown that solving for thin-plate splines through known points in two dimensions is equivalent to interpolating these points using the biharmonic radial basis function $\phi(r) = r^2 log(|r|)$ (fig.3.2(e)). In three dimensions, the thin-plate solution is equivalent to interpolating these points using the triharmonic function $\phi(r) = |r|^3$ (fig.3.2(f)).
Notice that this theory was developed in general dimension and a class of basis functions

was proposed in order to minimize (3.18)

$$\int_{\mathbb{R}^d} \sum_{|\alpha|=m} \frac{m!}{\alpha!} (D^\alpha f)^2 d\mathbb{R}^d. \tag{3.18}$$

Using this kind of basis functions (3.17), the corresponding matrix $G_{X,\Phi}$ naturally define quadric form :

$$Q_{X,\phi} : (\alpha_1, \cdots, \alpha_n) \longmapsto \alpha^T G_{X,\phi} \alpha \tag{3.19}$$

on $\mathbb{R}$ but these forms are positive definite only on a proper subspace of $\mathbb{R}^n$

To handle this problem, a multivariate polynomial q can be added to the weighted sum (3.20) in order to ensure positive definiteness of the solution, i.e the space of admissible function is augmented by the space, $\mathbb{P}_m^d$, of polynomial of order up to m.

$$f(x) = \sum_{j=1}^{n} \alpha_j \phi(\|x - x_j\|) + q(x) \quad q \in \mathbb{P}_m^d \tag{3.20}$$

Let $\{q_k\}_{k=0 \ldots l}$ be a basis of $\mathbb{P}_m^d$, thus f is expressed as

$$f(x) = \sum_{j=1}^{n} \alpha_i \phi(\|x - x_j\|) + \sum_{k=1}^{l} \beta_k q_k(x). \tag{3.21}$$

Notice that the interpolation problem (3.10) give us n equation with $n + dim(\mathbb{P}_m^d)$ unknowns.

Thus, extra equation are necessary to cover the right number of variables and unknowns. These equations will come from requiring polynomial reproduction. That is, if the data come from a polynomial $q \in \mathbb{P}_m^d$, the interpolant must coincide with q, i.e $f(x) \equiv q(x)$ (see proofs in [?, Buh03]).

This condition requires additional equations involving α as well as some conditions on $G_{X,\phi}$ itself [Buh03] and leads to a class of functions: *conditionally definite positive.*

Definition 3.4. *let $\phi : \Omega \times \Omega \to \mathbb{R}$ is a* ***conditionally positive definite function*** *of order m on $\Omega \subset \mathbb{R}^d$, iff for any choice of finite subsets $X \subset \Omega$ of n different points the value*

$$\alpha^T G_{X,\phi} \alpha = \sum_{j,k=1}^{n} \alpha_j \alpha_k \phi(x_j, x_k) \tag{3.22}$$

of the quadric form (3.19) is positive, provided that the vector $\alpha = (\alpha_1, .., \alpha_n)$ has the additionnal property

$$\sum_{j=1}^{n} \alpha_j q(x_j) = 0 \ \forall q \in \mathbb{P}_m^d \tag{3.23}$$

where $\mathbb{P}_m^d$ is the set of d-variate polynomials p of order up to m.

Let ϕ be conditionally definite positive of order d, The function f defined in Eqn. (3.21) is the unique solution to the interpolation problem:

$$\begin{cases} \sum_{j=1}^{n} \alpha_i \phi(\|x_i - x_j\|) + \sum_{k=1}^{l} \beta_k q_k(x_i) \quad \forall i = 1 \dots n \\ \sum_{j=1}^{n} \alpha_j q_k(x_j) = 0 \quad \forall k = 0..l \end{cases} \tag{3.24}$$

Thus the interpolation system (3.24) can be expressed with matrix as

$$\begin{pmatrix} G_{X,\phi} & Q_X \\ Q_X^t & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix} \tag{3.25}$$

where $G_{X,\phi}$ is the same matrix as (3.5), $\alpha = [\alpha_i]_{i=1\dots n}$, $\beta = [\beta_k]_{k=1\dots l}$ and

$$\mathbf{Q_X} = \begin{pmatrix} q_1(x_1) & q_2(x_1) & \dots & q_l(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ q_1(x_n) & q_2(x_n) & \dots & q_l(x_n) \end{pmatrix} \tag{3.26}$$

Proof: The system (3.25) is uniquely solvable.
Let (α, β) be a pair of vector that solve the homogeneous system with matrix (3.16). We have $G_{X,\phi}\alpha + Q\beta = F$ and $Q^t\alpha = 0$.
Multiplying the first with α^t, we obtain $\alpha^t G_{X,\phi}\alpha + \alpha^t Q\beta = 0$.
Transposing the second equation we obtain $\alpha^t Q = 0$.
Thus $\alpha^t G_{X,\phi}\alpha + 0 = 0$ and from the conditional positive definiteness of $G_{X,\phi}$ we deduce $\alpha = 0$.
We are left with $Q\beta = 0$. As X is $\mathbb{P}_m^d$-nondegenerate, this implies $\beta = 0$.
$\square$

Examples of radial basis functions

- *Linear* (fig.3.2(a)):

$$\phi(r) = r \tag{3.27}$$

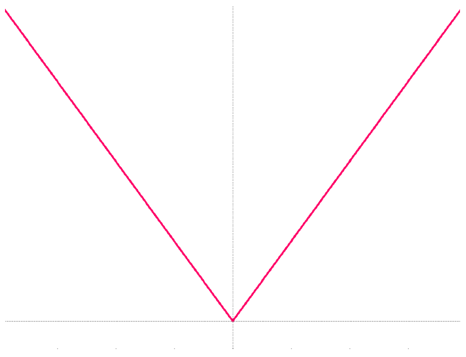
  Notice that the linear (3.27) basis function corresponds, in one dimension, to the piecewise linear interpolation that is the simplest case of spline interpolation.

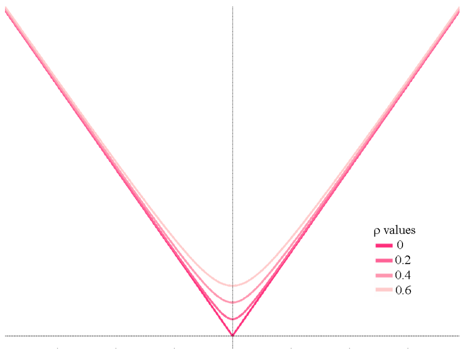
- *Multiquadrics* (fig.3.2(b)):

$$\phi(r) = \sqrt{r^2 + \rho^2} \qquad \rho \in \mathbb{R} \tag{3.28}$$

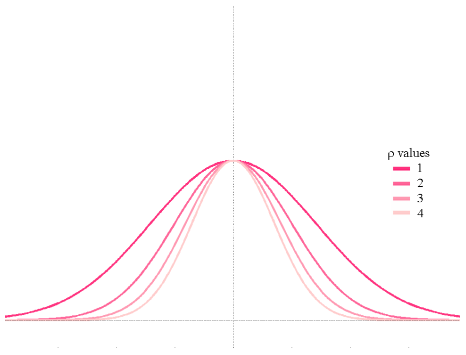
- *Inverse multiquadrics* (fig.3.2(d)):

$$\phi(r) = \frac{1}{\sqrt{r^2 + \rho^2}} \qquad \rho \in \mathbb{R} \tag{3.29}$$

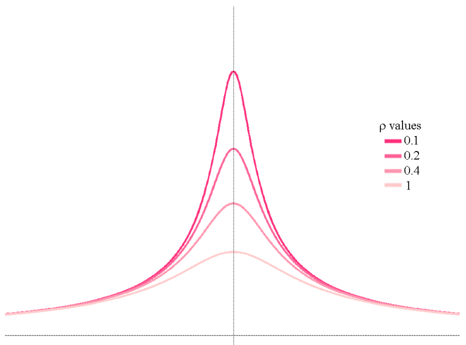

(a) Linear function:$\phi(r) = r$



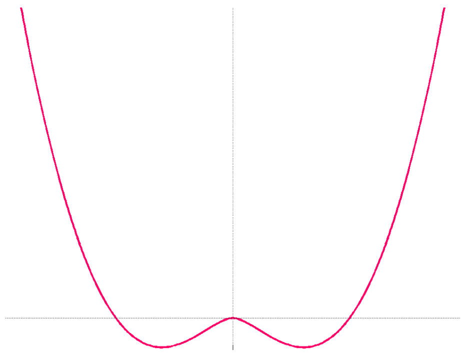
(b) Multiquadric: $\phi(r) = \sqrt{r^2 + c^2}$



(c) Gaussian function $\phi(r) = e^{-\rho * r^2}$



(d) Inverse multiquadric $\phi(r) = \frac{1}{\sqrt{r^2+\rho^2}}$


(e) Thin plate spline: $\phi(r) = r^2 log(r)$

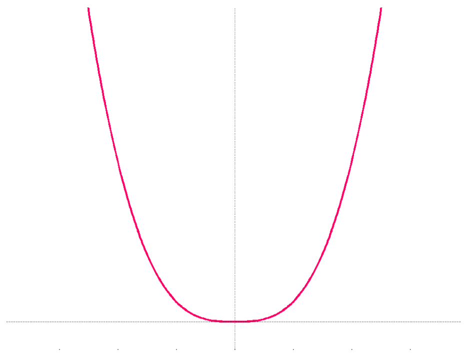

(f) Triharmonic: $\phi(r) = r^3$

Figure 3.2: Different class of RBF

- *Gaussian* functions (fig.3.2(c)):

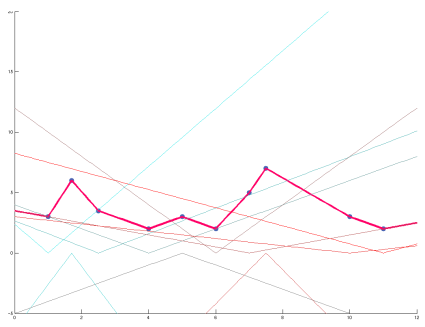
$$\phi(r) = e^{-\rho * r^2} \qquad \rho \in \mathbb{R} \tag{3.30}$$

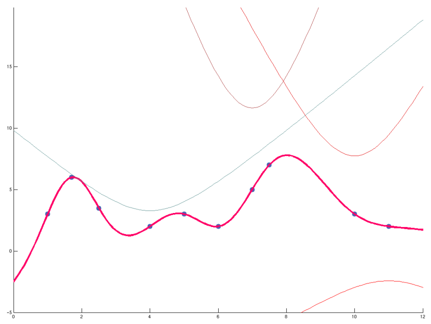
(the common basis function in neural network).

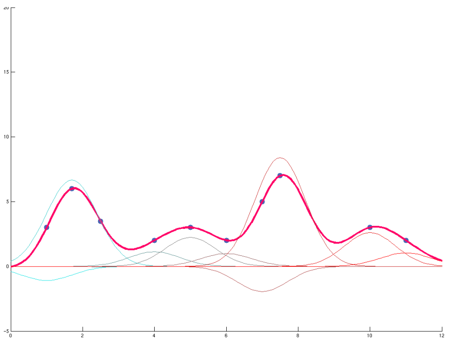
The inverse multiquadrics and the Gaussian function share a common property: they are both localized function, i.e. with a bounded support (fig.**??**).

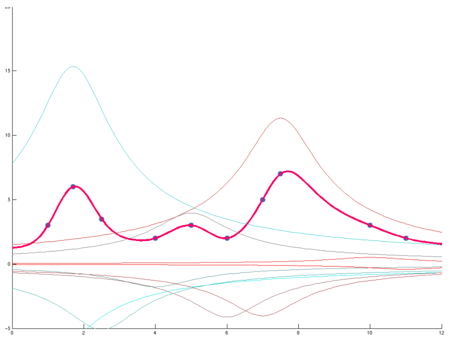
Notice that the RBF width are usually fixed to some value which is proportional to the max between the chosen centers.

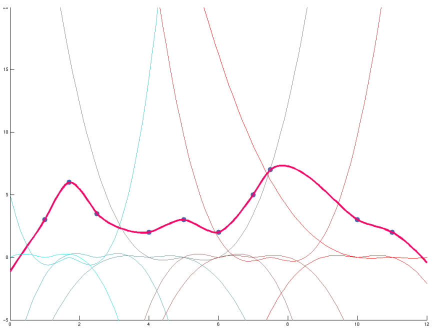
The figure 3.3 show different reconstruction of the same point set using the different class of RBF described above.


(a) Linear function


(b) Multiquadric


(c) Gaussian


(d) Inverse multiquadric


(e) Thin Plate function

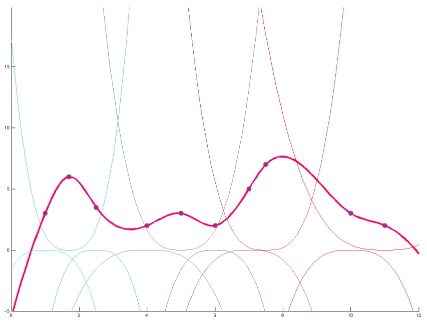

(f) Triharmonic

Figure 3.3: Curve Reconstruction using different basis functions. For each different class of RBF, the same set of blue points is reconstructed. The resulting red curve is constructed as a mixture of the basis functions.

3.3 RBF Surface reconstruction

The RBF approach approximates the surface S as the zero-level set of a function f (see sec.1.1.2). The reconstructed surface S' is defined as

$$S = \left\{ x \in \mathbb{R}^3 : f(x) = 0 \right\} \tag{3.31}$$

where $f(x)$ is expressed as a weighted sum of basis function ϕ centered on some points c_j called centers. Thus the equation (3.3) become

$$f(x) = \sum_{j=0}^{n} \alpha_j \phi(\|x - c_j\|) \tag{3.32}$$

Input: $P = \{p_i\}_{i=1\ldots n}$ a set of n points measured on the surface S.

3.3.1 Constraints

By definition, the function f should vanish on points of ^{S}. In absence of noise, as the points p_i are measured on the surface S, all the f_i are fixed to 0. Thus, the input of the reconstruction algorithm is the set of constraint points P associated with n scalar values $\{f_i\}$ such that $f_i = 0$.

The reconstruction then deal with solving the minimizing the energy (3.1) reformulated as follow :

$$E(f) = \sum_{i=1}^{n} \left(f_i - \sum_{j=0}^{m} \alpha_j \phi(\|p_i - c_j\|) \right)^2 . \tag{3.33}$$

To avoid the trivial solution $\alpha = \overrightarrow{0}$, several interior and exterior constraints are added where the function f does not vanish (fig.3.3.1). The additional constraints are called *off-constraints*. For each off-constraint $q_k \in Q$, we assign to f a signed value $f_k = \pm d(q_k, P)$,

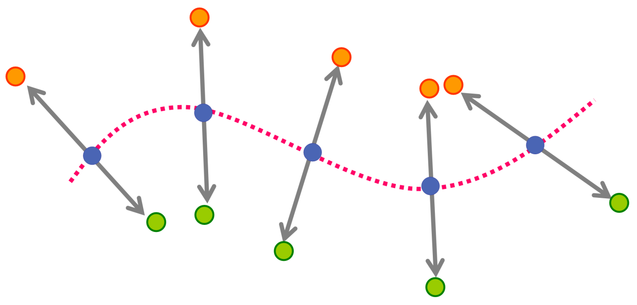


Figure 3.4: Additional off-surface points along the surface normals.

where $d(q_k, P)$ is the distance from q_k to the point set P. The N constraints $X = \{x_i\}_{i=1\ldots N}$ are now composed of the set P, the n input points, and the set Q, the off-constraints. Typically, 2 off-constraints are added for each constraint $p \in P$, thus $N \approx 3n$.

Remarks: normals to the surface S need to be known in order to compute the off-constraints.

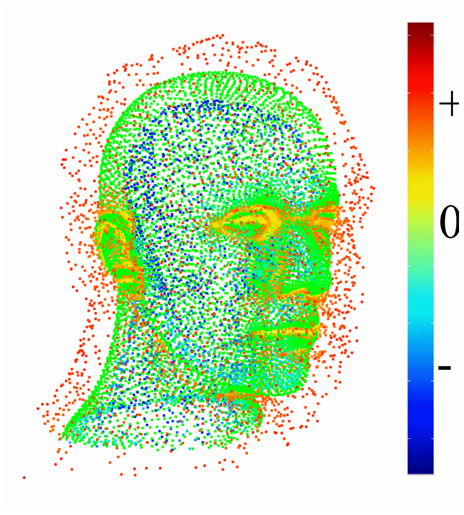


Figure 3.5: All the constraints points. The colors represent the function values (cold tones for negative, hot tones for positive values and green for the zero values).

3.3.2 Centers

Most approaches locate the centers at the constraints points, therefore the number of centers is such that $m = N$.

$$c_i = x_i \qquad \forall i = 1 \dots N \tag{3.34}$$

The minimization process (3.1) reduces to solving a $N \times N$ linear system which requires $O(N^3)$ machine operations and $0(N^2)$ bits for storage. Then, each evaluation of $f(x)$ requires $O(N)$ operations. This approach is therefore not suitable to a number of constraints greater than several thousands. To reduce the computational complexity, one first idea is to reduce the number of constraints. Notice that since most algorithms use the same points as constraints and as centers, this also leads to center reduction. This approach is commonly called *center reduction* in the literature.

3.3.2.1 Center Reduction

Some constraint points are relevant while some others are redundant or even completely irrelevant. Therefore deal with fewer constraints points is useful.
Constraint reduction consists in approximating, with the desired accuracy, all input data using fewer constraints but significant constraints. If the function f are defined to be centered on the constraint points, then reducing the constraints is equivalent as reducing the centers.

A greedy algorithm is proposed in [CBC+01]: centers are iteratively added at locations where the fitting error is maximum until a satisfactory accuracy is reached.

The interpolation system (3.11) is solved using only the selected centers/constraints and the fitting accuracy is evaluated on all the input points.

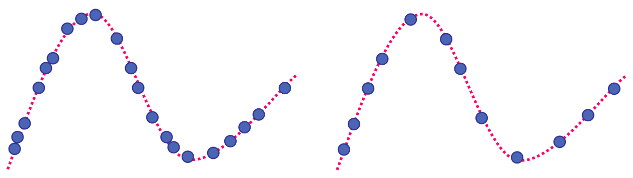


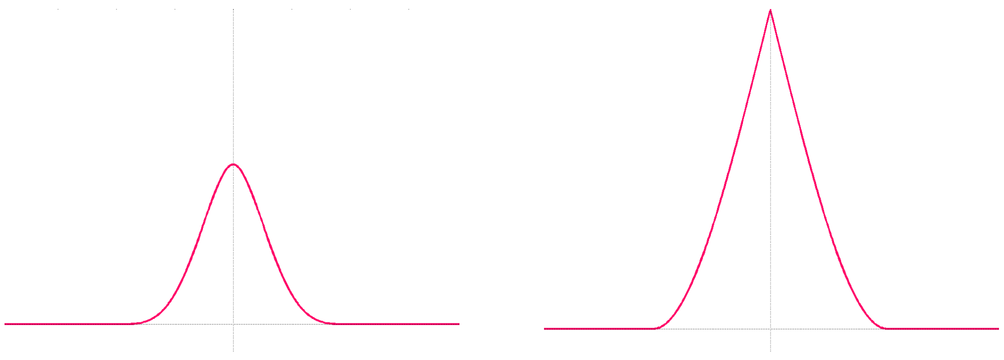
Figure 3.6: Center reduction as performed by [CBC+01].

3.3.3 Basis functions

All functions listed in the section **??** have a unbounded support. The corresponding equations lead to a dense linear system, therefore recovering a solution is tractable only for small data sets. In order to obtain a sparse interpolation matrix, Compactly Supported RBFs (CSRBF) have been introduced by Wendland in [Wen95]. A lot of CSRBFs used for reconstruction was proposed in the literature [Sch95, Wu95]. Note that these basis functions are not suited for the reconstruction from inhomogeneously sampled surfaces. Here follow two examples of CSRBF (Fig. 3.7):

- Wu function $\phi(r) = (1-r)_+^2(2+r)$
- Wendland function $\phi(r) = (1-r)_+^4(1+4r)$

where the symbol "+" denotes $(x)_+ = x$ if $x > 0$ else $(x)_+ = 0$

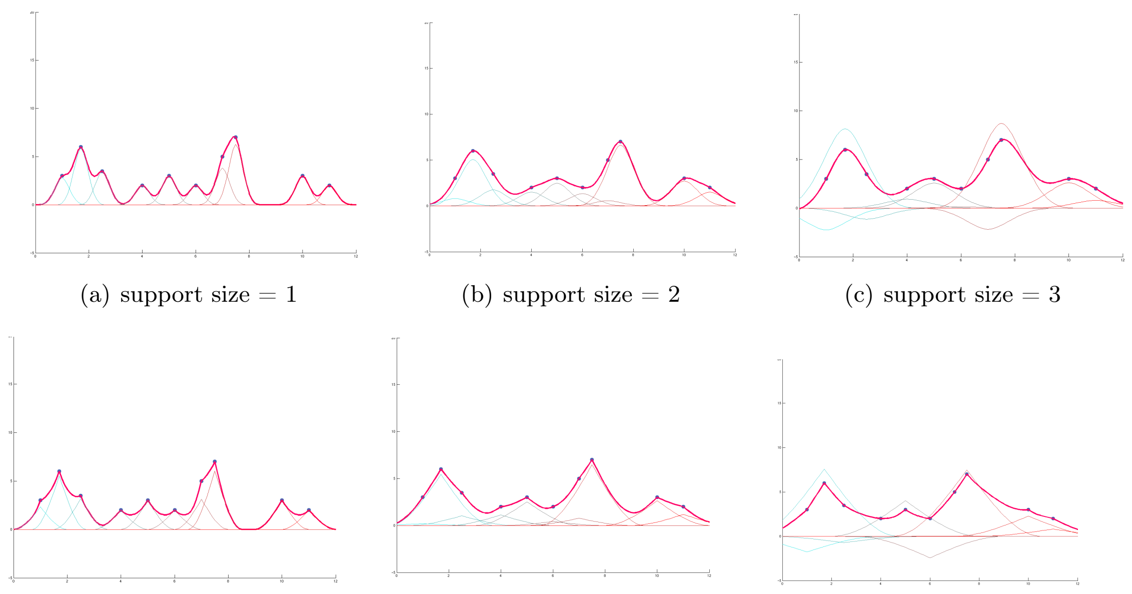


(a) Wendland function :$\phi(r) = (1-r)_+^2(2+r)$ (b) Wu function : $\phi(r) = (1-r)_+^2(2+r)$

Figure 3.7: Two compactly supported RBF.

The size of the support is important. Besides acting on the sparsity of the matrix, it infer on the property of the reconstructed function. The lager the support, the smoother the function, as it shown on the figure 3.8, but the more dense the matrix. Trade smoothness for the sparsity of the matrix must be found. Typically, the support size is the same for all the basis function.

Ohtake [OBS04] proposed to locally adapt the support of the basis functions. A support σ_j is then be associated at each center c_j. This approach is done in addition to a center selection procedure according to the support σ_j. The selection is made in order to have an amount of overlap of the cover less than a certain threshold.



(a) support size = 1 (b) support size = 2 (c) support size = 3

(d) support size = 1 (e) support size = 2 (f) support size = 3

Figure 3.8: Curve Reconstruction using CSRBF with different supports size. The same set of blue points is reconstructed. The resulting red curve is constructed as a mixture of basis functions. Top : Wendland function. Bottom: Wu function.

We can notice that radially symmetric functions are not suited to piecewise smooth surface reconstruction. Dinh et al. [DTS00] tryed to overcome this issue by using anisotropic basis functions. But we can notice that as the basis function are smooth, the function solution is smooth too even in case of extreme anisotropy. As we know, the only approaches which reconstruct piecewise smooth surfaces are based on a decomposition of the space.

3.3.4 In practice

The RBF approach is quite simple in theory, it can be summarized by solving a linear system, however the system is large in practice. The naive approach requires $O(N^3)$ machine operations to solve the system and $0(N^2)$ bits for storage. Several strategies was proposed: data reduction, for example the *center reduction* (sec.3.3.2.1), or making the matrix sparse with compactly supported RBF (sec.3.3.3). In the following, we present some other approaches for fast evaluation with dense matrix or fast reconstruction based on space decomposition techniques.

3.3.4.1 Evaluation

When the input point set is large but sparse, CSRBF may not well suited. Thus polyharmonic basis functions must be used. In the case of globally supported basis functions, like polyharmonic RBF, m evaluation of $f(x)$ requires $O(mN)$ operations, where N is the number of constraints.

In [CBC+01] Carr and Beatson proposed to performs fast evaluation using Fast Multipole Method (FMM). This algorithm reduce the computational cost of the evaluation: a reconstructed function, which is the sum of N polyharmonic radial basis function, is evaluated at $m \geq N$ points with $O(m + N \log N)$ operations.

The main idea of this approach is the fact that when computation are performed an exact precision is neither required nor expected. Thus, approximations based on *far field and near field expansions* are performed. The centers are clustered, and for a given evaluation point x, the evaluation is approximate in the clusters far from x and is computed directly for clusters near from x.

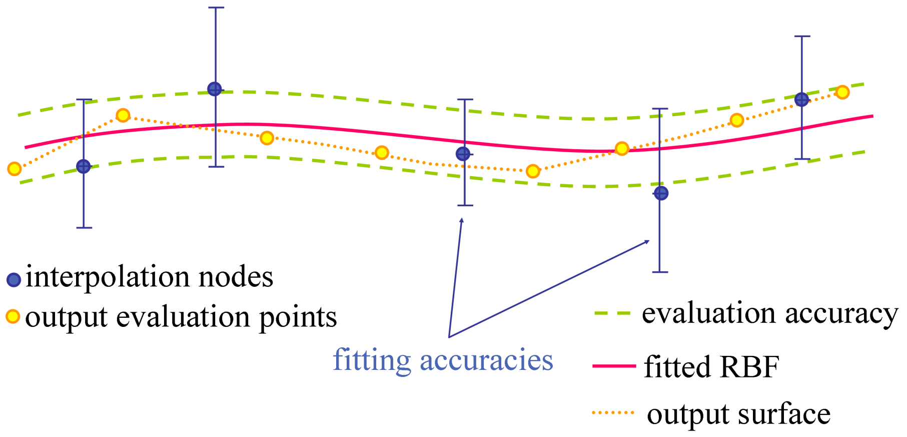


Figure 3.9: Illustration of fast evaluation. A RBF is evaluated at regular intervals lying between the dashed evaluation accuracy bands either side of the actual function.

To perform the approximations two parameters were introduced: a fitting accuracy and an evaluation accuracy. The fitting accuracy specifies the maximum allowed deviation of the fitted RBF value from the specified value at the interpolation nodes. The evaluation accuracy specifies the precision with which the fitted RBF is then evaluated.

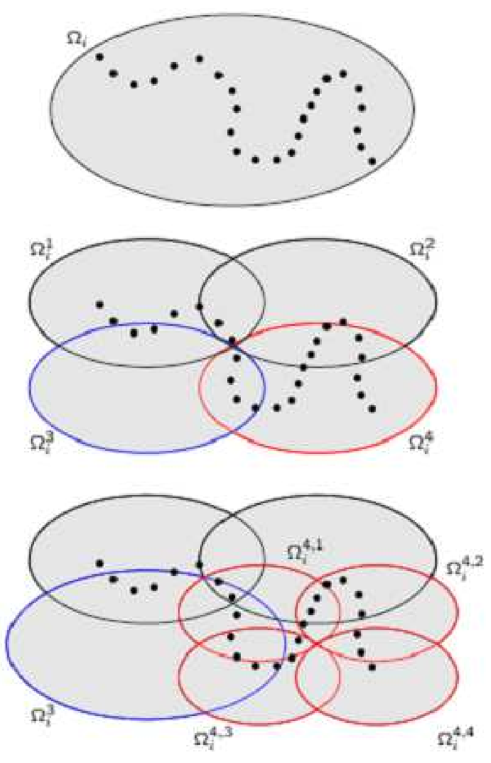
Remarks: the development of the FFM methods is very complex.

3.3.4.2 Domain Decomposition and Multilevel Methods

The original data set is subdivided into several smaller data set. The problem is then solve iteratively on each cluster. In addition to allow piecewise smooth surface reconstruction, domain decomposition approaches allow to deal with very large point set and to perform local reconstruction. At last, the underlying data structure give a multiscale representation.

Partition of Unity Wendland, in a theoritical survey [Wen02], combine the Partition of Unity (see sec.2.2.3) method and the radial basis functions. Then, Tobor et al. [TI04] proposed an efficient algorithm based on this idea.

Let Ω be a bounding box of the input point set. Ω is divided into M "slithly" overlapping subdomains $\{\Omega_i\}_{i=1...M}$ such as $\Omega \subset \cup_i \Omega_i$ (fig.3.10). On each subdomain Ω_i, a local RBF reconstruction is performed, i.e. a function f_i is computed. Then, the global solution



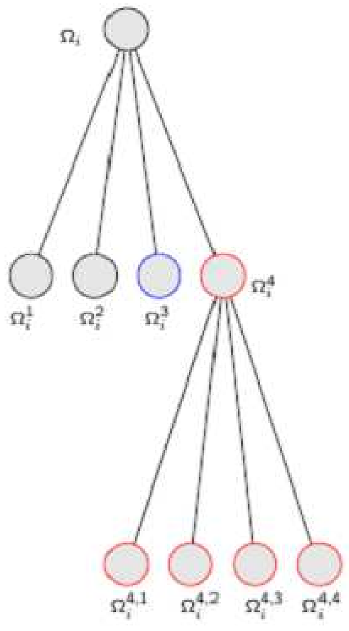


Figure 3.10: Partition of unity. Left: The space is subdivided according to the number of points in each cell. Right: the tree corresponding to the different levels which the leaf are the final subdomains.

f is constructed by blending together all the local function f_i, thus f is defined as

$$f(x) = \sum_{i=1}^{M} f_i(x) w_i(x) \tag{3.35}$$

where the w_i are weight function. In order to perform a partition of unity algorithm, the weight must verify the condition $\sum_{i=1}^{M} w_i = 1$. This condition is obtained from any other set of smooth functions W_i by a normalization procedure:

$$w_i(x) = \frac{W_i(x)}{\sum_{j=1}^{M} W_j(x)} \quad \forall i = 1 \dots M \tag{3.36}$$

The W_i functions are defined as the composition of a distance function and a decay function.

This algorithm can be mixed with a greedy selection of the center: a notion of residual and a root-mean-square error is introduced in order to evaluate the fitting accuracy. Then new centers are added in subdomains where the error is too important. Thus, although it is not explicitly a multiresolution method, this approach could be used to establish a multiscale representation by using the intermediate solutions.

Multilevel Adaptive CSRBF Othake et al [OBS03, OBS04, **?**] presented a multilevel and adaptive method using compactly supported RBF which the support size is defined locally.

Given a set of n input point P equipped with normals. The centers, $C = \{c_i\}_{i=1..m}$ are chosen among P such that $m < n$.

Single level approximation: This approach is not purely RBF based. The reconstruction is made in 2 steps. First, a local approximation by least square fitting is performed, then a radial basis function algorithm is done to recover the small details.

The function solution f is formulated as

$$f(x) = \sum_{c_i} [g_i(x) + \lambda_i]\phi_{\sigma_i}(\|x - c_i\|) \tag{3.37}$$

where the unknowns are the function g_i and the weights λ_i. The g_i function is a local approximation of P in $\{x \in \mathbb{R}^3 | \|x - c_i\| \leq \sigma_i\}$ a small neighborhood of c_i. Then we determine the set $\{\lambda_i\}$ from m interpolation conditions:

$$f(c_i) = 0 \tag{3.38}$$

The zero-level set of $\sum_{c_i} g_i(x)\phi_\sigma(\|x - c_i\|)$ approximate P. For each c_i, the ϕ_{σ_i} function, centered on p_i, is compactly supported with a support size scalable according to the center density in a neighborhood or c_i.

$$\phi_{\sigma_i}(x) = \phi(\frac{x}{\sigma_i}) \quad \forall i = 1 \ldots n \tag{3.39}$$

The equation (3.37) can be rewritten as

$$f(x) = \sum_{p_i} g_i(x)\phi_{\sigma_i}(\|x - c_i\|) + \sum_{p_i} \lambda_i \phi_{\sigma_i}(\|x - c_i\|) \tag{3.40}$$

The first term of the right-hand side of (3.40) can be considered as a base approximation and the second term represents local details.

The centers c_i and their influence region determined by σ_i are selected in order to obtain a "good" covering of the data points with an amount of overlap greater than a certain threshold. A confidence value on each data point is also used. This confidence value is an input of the algorithm.

Multilevel approximation: Using an octree, the point set P is clustered. At each cell corresponds a basis function, i.e. a center which is the centroid of the points in the cell.

Then a coarse to fine reconstruction approach is performed. The function f (3.37) at the level k is computed according to the function computed at the level $k-1$:

$$f^k(x) = f^{k-1}(x) + o^k(x) \tag{3.41}$$

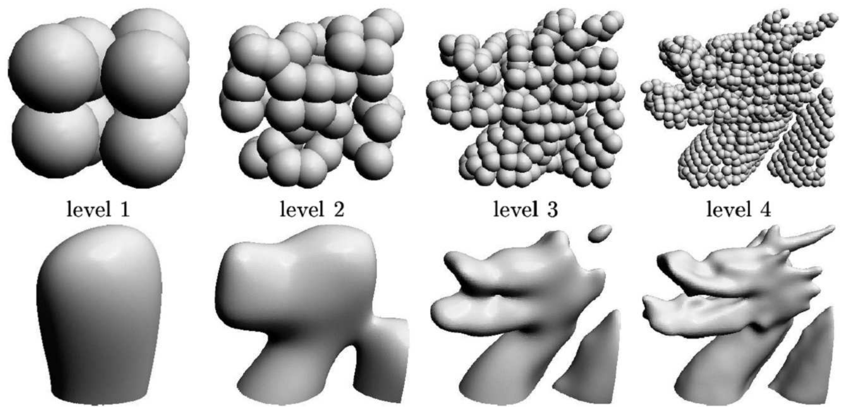


Figure 3.11: Multi-scale interpolation of the Stanford dragon model. from left to right: four first levels of the multi-scale hierarchy. Top row: the spheres corresponding to the support of the rbf. Bottom row: the zero level set of the interpolating function.

where $f^0(x) = -1$ and o^k is an offsetting function, the residual of the $k-1$ level reconstruction:

$$o^k(x) = \sum_{p_i^k \in \mathcal{P}^k} [g_i^k(x) + \lambda_i^k]\phi_{\sigma_i}^k(\|x - p_i^k\|) \tag{3.42}$$

$\mathcal{P}^k$ is the set of centers at the level k approximated by f^k . In the first level, $\mathcal{P}^1$ corresponds to the subdivision of the bounding box into height equal octant, i.e $\mathcal{P}^1$ contains height centers. $\mathcal{P}^k$ is obtained by subdividing each leafs of $\mathcal{P}^{k-1}$

3.4 Generalized Radial Basis Functions

All the previous approaches locate centers both at the input data points and at the off-surface constraints.The main advantages of these methods are that the matrix is squared, symmetric and that with "small" additional constraints there is an unique solution.

Another idea to further reduce the number of centers while maintaining decent fitting accuracy is to relax the one-to-one correspondence between the centers and the constraints and their localization. This approach is called *Generalized Radial Basis Functions* (GRBF) in the neural networks community [PG89a, BL88, PG89b].

Let m be a user-defined number of centers, possibly located anywhere in space, and N the number of constraints, such that $m << N$. The function f can be expressed as:

$$f(x) = \sum_{j=1}^{m} \alpha_j \phi(\|x - c_j\|), \tag{3.43}$$

and thus the matrix of the least-squares system (5.9), with size $N \times m$, is formulated as follows:

$$G_{X,C,\phi} = \begin{pmatrix} \phi(\|x_1 - c_1\|) & \phi(\|x - 1 - c_2\|) & \dots & \phi(\|x_1 - c_m\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|x_N - c_1\|)) & \phi(\|x_N - c_2\|) & \dots & \phi(\|x_N - c_m\|) \end{pmatrix} \tag{3.44}$$

Therefore, the size of the matrix to be inverted and stored is now $m \times m$, independently of the number of constraints. $O(m)$ operations are now required for a single point-wise evaluation.

It is possible to take into account as much as constraints we want, i.e as much information it desired. However, each term of the matrix $G^t_{X,C,\phi} G_{X,C,\phi}$ of the system (5.9) being a sum of contributions arising from each constraint, the number of constraints conditions the cost for assembling the matrix.

In the classical approaches using RBF, the degrees of freedom are the basis function (compactly supported or not, its order of continuity,...) and the number of centers, i.e. of constraints.

For the generalized RBF, there are additional degrees of freedom. Besides the number of centers, we can chose their location. Let m be the desired number of centers. In the neural network field, several strategies are proposed: the easiest way to chose m centers is to take randomly m points in the space. Alternatively, the m points can be selected after a clustering process on a set of candidate centers. We can note also the *Movable centers* [PG89a] approach which consists in determining centers adaptively with a gradient descent technique. In the same way, we can choses the constraints number and location apart from the centers.

In ours work, we exploit one of the most important degrees of freedom offered: the location of centers and constraints to obtained a satisfactory trade-off between number of centers and fitting accuracy. In addition to this choice, we propose use compactly supported RBF which the support is adapted to the local geometry of the point set P. Our aim being to obtain a compact representation with a reconstruction algorithm with few parameters.

Part III

Contribution: Voronoi Centered Radial Basis Functions

Chapter 4

Introduction

Although Voronoi-based reconstruction has long been criticized for its computational burden, recent developments in the implementation of fast algorithms have alleviated this issue. As an example, computing the Delaunay of 50K points takes 1s using the CGAL library [FGK+00]. The efficiency and the accuracy theses methods still depend heavily on the quality of the sampling and on the differential and topological properties of the surface. In particular, sparsity, redundancy, noisiness of the sampling, or non-smoothness and boundaries of the surface makes the surface reconstruction a challenging problem. Besides, Voronoi-based reconstruction methods generally fail to produce watertight surfaces.

Radial basis functions, on the other hand, still have issues with picking the right non-zero constraints (to avoid disconnected components), and with efficiently computing the weights. Functions with unbounded support give the best reconstruction results, but also lead to dense matrices. The only viable solution to this problem so far is the multipole expansion for polyharmonic functions developed by Carr et al. [CFB97]. Unfortunately this approach is notoriously intricate and difficult to reproduce. Compactly supported functions lead to sparse matrices [Wen95]. However, finding a proper support size for the functions in case of irregularly sampled surfaces is difficult. Besides, when the basis functions is compactly supported, the computed function is only defined in the vicinity of the input data points.

A recent trend is to perform a set of local reconstructions, which may be mixed with quadric or higher-order jet fitting, and to blend them using the partition of unity [TI04, OBS04]. Although a great deal of effort has been put into the elaboration of multi-level techniques with local reconstructions to deal with large data sets, less effort has been spent at improving the compactness of the representation by center selection and optimization [CFB97, TI04, OBS04].

4.1 Contributions

Our approach combines both worlds of Voronoi based and radial basis function reconstruction and eliminates some of the aforementioned shortcomings. The sampled surface S is still reconstructed as the zero-level of a function f expressed as a linear combination of radial basis functions. The main advance in our method is to use radial basis functions centered at vertices of the Voronoi diagram of the data points. More precisely, the centers of the radial basis functions are chosen among a subset of those Voronoi vertices, which are called poles. Under certain sampling conditions, the poles are known to be closed to the medial axis of the sampled surface S [AB98]. Furthermore, each pole is the center of a Delaunay ball hereafter called polar ball. A polar ball is a maximal ball empty of sampled points. Such a ball is close to a maximal ball in $\mathbb{R}^3 \setminus S$. Considering that any smooth surface S can be viewed as the envelope of the maximal balls in $\mathbb{R}^3 \setminus S$, using poles as centers for radial basis functions is a rather natural idea. Furthermore, in our reconstruction process, we use the radius of each polar ball as a guidance for choosing the support of the corresponding basis functions. Hence, the support of each basis functions is locally adapted to the geometry and topology of the sampled shape. Also, because the radius of each polar ball is a good estimate of the distance between the pole and the sampled surface, we use this radius to set, as additional constraints, the value of the function f at the poles. This leads to a reconstruction technique with the following features:

- The surface is represented as the zero-level set of a signed function, which is a good approximation of the signed distance field to the surface.
- The function is defined as a weighted combination of locally supported radial functions; the number of basis functions is independent from the number of input points and typically significantly smaller. The function can thus be evaluated faster than when using traditional (even compactly supported) RBF.
- While the computation of the weights potentially takes into account all input data points as constraints, the size of the system matrix only depends on the number of centers, not on the number of constraints.
- A filtering of the poles based on the notion of λ-medial axis allows the surface to degrade gracefully with noise.

In comparison with Voronoi-based reconstruction, the most important advantages of our technique are the resilience to noise and the construction of a smooth watertight surface that approximates all data points. In comparison to the common compactly supported RBF, fewer centers are used for the same accuracy. This leads to faster computation of the weights and faster evaluation of the function. Using poles associated with their Voronoi ball radius as additional constraints leads to a better approximation of the distance field to the surface, and to fewer topological issues such as superfluous connected components away from the input points.

4.2 Overview

The problem to solve is the following: given a point set $P = \{p_i\}_{i=1..n} \subset \mathbb{R}^3$ measured of a surface S. We want to construct a surface S' that approximates S. The surface S' is defined as the zero level set of a function f. f is expressed as a weighted sum of compactly supported radial basis function. In our algorithm, we explore the degree of freedom of the generalized radial basis function approach: center number and location, constraints number and location and the basis functions.

The plan of this part is as follow. The chapter 5 contains an overview of our algorithm. In the chapter 6, we detail the different strategies for the centers choice. We then provide in section 7 the detail of the implementation of the constraints classification and the matrix construction. Finally, the chapter 8 contains several results which illustrate all our choices.

Chapter 5

Algorithm

Given a set of input points $P = \{p_i\}_{i=1..n} \subset \mathbb{R}^3$ measured on a surface S, we want to construct a surface S' that approximates S.
We recall that we restrict to the case where the surfaces divide the space into two subspace: a bounded volume tagged as inside and an unbounded volume tagged as outside.

Our approach is RBF based (see. chapter 3), the surface S' is defined as the zero level set of an unknown function f. f is defined as a weighted sum of radial basis functions:

$$f(x) = \sum_{j=0}^{n} \alpha_j \phi(\|x - c_j\|). \tag{5.1}$$

As the input points P are supposed to lie on the surface, the value f_i of the function f at the points p_i is set to zero (5.2):

$$f_i = f(p_i) = 0, \qquad \forall i = 1 \dots n. \tag{5.2}$$

Our aim is to find the α vector in (5.1) in order to minimize the energy functional:

$$E(f) = \sum_{i=1}^{n} \left(f_i - \sum_{j=0}^{m} \alpha_j \phi(\|p_i - c_j\|) \right)^2 . \tag{5.3}$$

The main idea of our algorithm is to use Generalized RBF (see section 3.4) with centering the ϕ function on some Voronoi Vertices. Our reconstruction must be adapted to the number of centers, m, fixed by the user.

Our algorithm proceeds as follows: we first compute the Delaunay triangulation of the input points and in the same time its dual, the Voronoi diagram. Our algorithm then refines a subset of the Voronoi vertices. In the first stage, poles are extracted from the Voronoi vertices and are classified as inside or outside. In the second stage, the m centers are selected to sample a part of the medial axis.The selection is performed by filtering, then clustering the set of poles or by selecting the pole with a greedy algorithm.

In the first case, the poles are filtered in order to adjust the level of detail to the budget of centers and clustered in order to achieve a center distribution nicely spread on the medial axis. In the second case, the m poles are selected in order to achieve at the same time a nice sampling of the medial axis adjusting the level of detail to the budget of centers.

We choose as radial basis function a Gaussian-like function with a compact support [Wen95], where the support size is locally adapted. As constraints, we impose the function f to be zero at the data points and to be non zero at the center points. A value set at a center point approximates the signed distance from this point to the sampled surface. The weights are obtained by computing the best least squares approximation of the values of the function f to data points and the constraint points.

We structure this section by the main components of the reconstruction algorithm, namely the choices made for the centers, for the constraints and for the radial basis functions. At last, we present the system to be solved.

5.1 Centers

The centers of the basis functions are selected from the vertices of the Voronoi diagram of the input points. We recall that Every sample point $p \in P$ generates a Voronoi cell and the vertices of the cell furthest away from p on the two sides of the surface are the poles of p. Each pole is the center of a Delaunay ball called *polar ball* (see sec.2.1.2). A polar ball is a maximal ball empty of sampled points. Such a ball is close to a maximal ball in $\mathbb{R}^3 \setminus S$.

The main idea is that a solids can be roughly approximated (exact in the limit) as a union of balls: the Medial Axis Transform.

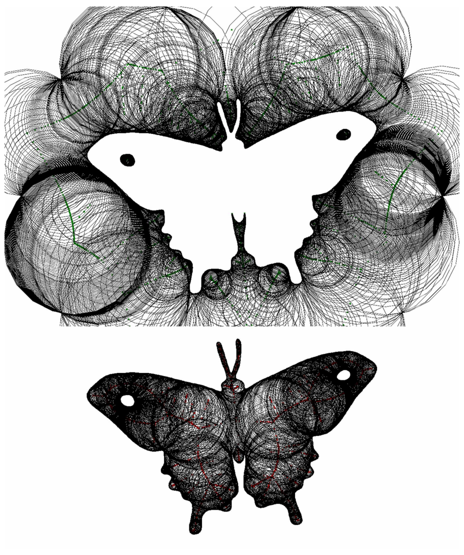
Definition 5.1. *The* ***medial axis transform*** *(MAT) of a smooth surface S is the representation of the surface by the set of maximal balls included in one of the two component of $\mathbb{R}^3 \backslash S$ (fig.5.1)*

Considering that any smooth surface S can be viewed as the envelope of the maximal balls in $\mathbb{R}^3 \setminus S$, using poles as centers for radial basis functions is a rather natural idea.

Let m be the user-defined budget of centers. Generally, the number of poles is greater than m, and we must select m relevant poles as centers in order to form a sampling of the medial axis, $M(S)$. There are two challenges to compute this sampling:

- the medial axis is highly unstable with respect to small details of the shape (fig.5.2);
- only a discreet approximation of the medial axis is known and its sampling is dependent of the distribution of the input points.

We thus construct the set of centers as a sampling of a part of the medial axis $M(S)$. Indeed, if m is small, there is no hope to reconstruct very small details and thus we need to remove the poles which correspond to the smallest details. Notes that small details are not distinguishable from noise. Furthermore, our sampling density had to be independent


(a) MAT in 2D. The outside and the inside component.

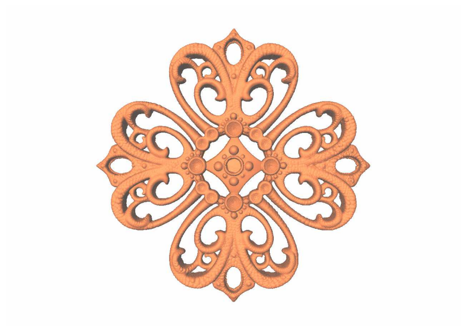

(b) MAT 3D. the inside component.

Figure 5.1: Medial Axis Transform (MAT)

from the distribution of the data points, thus we propose to perform a sampling according to a sizing field function.

The *sizing field function*, *ssf*, is defined on the medial axis. The sizing field function, *sff*, is constructed by associating at each point of the medial axis of the surface S, $M(S)$, the radius of the maximal ball centered on it. This function is continue on each component of the medial axis.

In the chapter 6, the selection of the m centers is detailed.

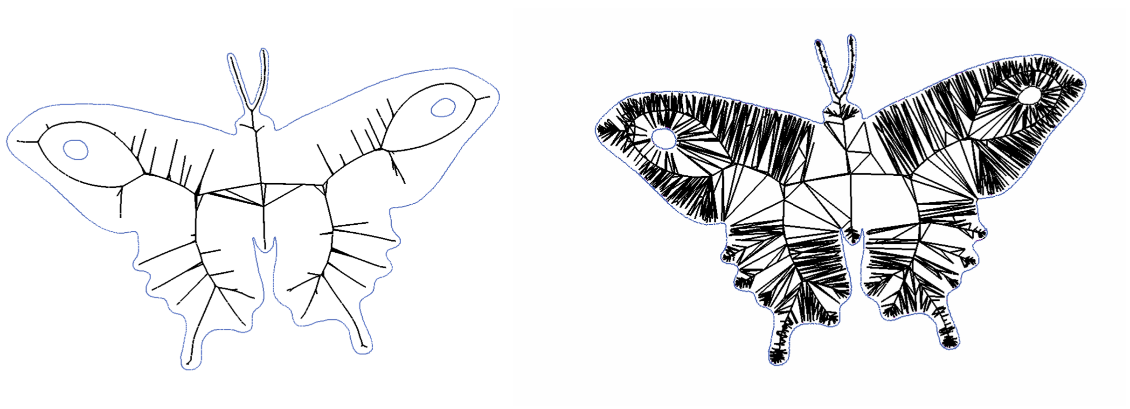


Figure 5.2: Instability of the medial axis. Left: a smooth surface and its inside medial axis (black). Right: the same surface with noise added and its (unstable) inside medial axis.

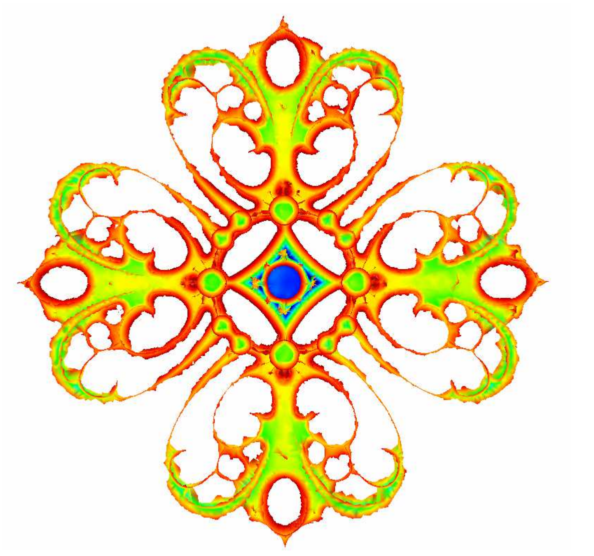


Figure 5.3: Sizing field on medial axis in 3D. The colors represent the sizing field values; cold tones for the minimum and warm tones for the maximal ones)

5.2 Constraints

We take as constraints both the input points where the function f is specified to be zero, and a set of additional constraints where f is specified to be non-zero. Recall that our goal is to consider as an approximation of the shape the zero-level set of f. Therefore, we wish to define a signed function f which is positive outside the shape, negative inside and with a non-zero gradient close to the sampled surface. A good candidate is a function approximating the signed distance function to the sampled shape where the distance is positive for points outside the shape and negative inside (Fig.5.4).

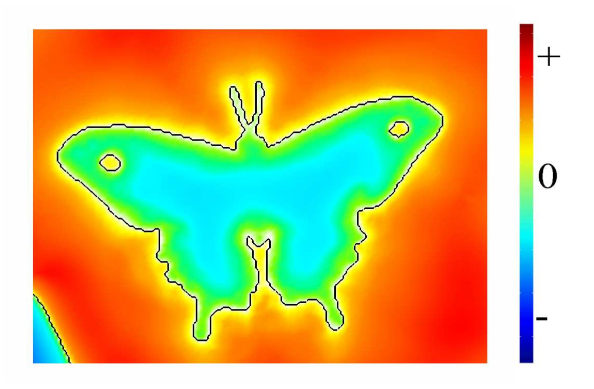


Figure 5.4: 2D shape (black) and the computed function. Colors range from cold color tones for positive distance values to hot color tones for negative distance values.

As we have seen in the section 2.1.2, the poles are shown to exhibit interesting properties:

- if v_i is a pole of the cell $V(p_i, P)$, the direction $\overline{v_i p_i}$ is a good approximation of the normal at p_i;

- the radius of the Delaunay ball centered at v_i is a good approximation of the distance from v_i to the sampled surface.

Thus, poles can be used as a constraints in order to approximate a distance function to the sampled surface. It remains however to determine the sign of this value, and therefore to *classify* the poles as inside or outside.

Pole Classification Pole classification is the process of labeling the poles as inside or outside the surface. Common approaches use an algorithm to propagate the pole labels through the graph built from adjacency relationships between the poles. In our implementation, we classify the poles using a variant of the algorithm proposed by Amenta [ACK01]. This variant, due to F.Cazals (internal communication), is more efficient and more robust against to noise. During the classification process, a location tag (inside, outside and undetermined) and a confidence value are attributed to each pole. If the confidence of a pole is lower than a certain threshold, the pole will not be taken into account as a constraint. The algorithm is detailed in the section 7.1.

5.3 Basis Functions

Recall the reconstructed surface is required to be independent of Euclidean transformation. The function Φ is thus restricted to the set of radial functions (see sec.3.2):

$$\Phi(x, c_i) = \phi(\|x - c_i\|) \tag{5.4}$$

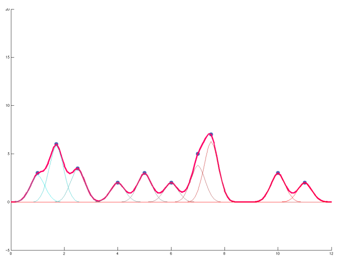
where $\|.\|$ denotes the Euclidean distance and $\phi : \mathbb{R}^+ \rightarrow \mathbb{R}$.

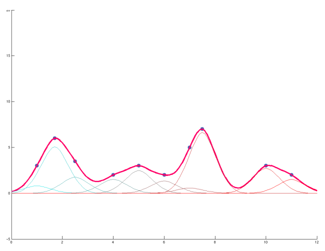
We chose to use a Gaussian-like function with a compact support [Wen95] where the support size is locally adapted.
As illustrated by figures section 3.3.3, the support size of the basis function impact on the reconstruction result (fig.5.5). The support size allow to adjust the reconstruction to the geometry of the curve (continuity, connexity,....).

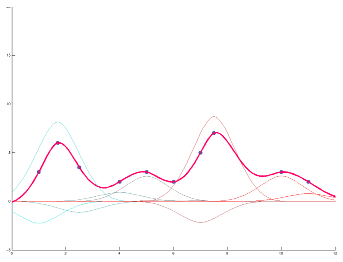
In the examples figure 5.5, the support is global thus it is difficult adapt the reconstruction to the local geometry of the shape. Different local supports can be used to handle this problem (fig.5.5(d)).

As centers are poles, each center c_i, has a corresponding to a scalar value, r_i, the radius of its polar ball. Our function of choice ϕ is compactly supported, and the support size s_i for the function centered on c_i is computed using to r_i. The ϕ function (5.4) centered on c_i is scaled according to the local support s_i:

$$\phi_i(\|x - c_i\|) = \phi(\frac{\|x - c_i\|}{s_i}). \tag{5.5}$$


(a) support size = 1


(b) support size = 2


(c) support size = 3

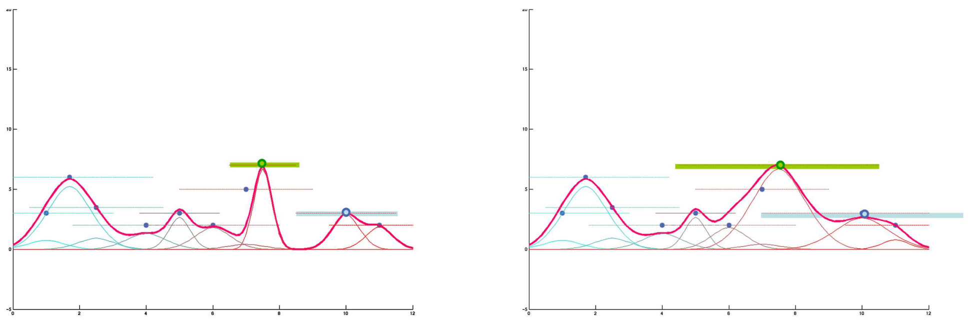

(d) variable size of local supports

Figure 5.5: The set of blue points is reconstructed using the Wendland functions (red curve). Top: For each figure the size of the supports changes but the support is the same for all the basis functions. Bottom: For each figure the size of the supports are the same excepted for two of the points where the support grow.

We want to use a class of radial basis function ϕ_j compactly supported in order to obtain a sparse matrix and to perform efficient evaluations. Furthermore, we want a class of smooth basis function since the basis function determine the smoothness of the approximant. A family of piecewise polynomial functions with local support was proposed

$$\phi(r) = \begin{cases} P(r) & \text{if } \; 0 \leq r \leq 1 \\ 1 & \text{if } \; r \geq 1 \end{cases} \tag{5.6}$$

where p denotes a univariate polynomial. Wendland have proposed several basis functions of minimal degree. We chose for our implementation the following function:

$$\phi(r) = (1 - r)_+^4 (1 + 4r), \tag{5.7}$$

where the symbol $+$ means $(x)_+ = x$ if $x > 0$ and $(x)_+ = 0$ otherwise. This function is smooth (C^2) and Gaussian like (Fig.5.5).

5.4 System Resolution

The centers are a set $\{c_j\}_{j=1\ldots m}$ of m points in $\mathbb{R}^3$. The constraints are the set $\{p_i\}_{i=1\ldots N}$ of N points where the value of f is known.

To solve the reconstruction problem, the energy (5.8) is minimized.

$$E(f) = \sum_{i=1}^{n} (f_i - f(x_i))^2 . \tag{5.8}$$

Thus the problem consists in solving a linear system with least-squares technique, i.e. to determine the vector $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ by solving a linear system

$$G^t_{X,\Phi} G_{X,\Phi} \alpha = F \tag{5.9}$$

where G be the matrix $[\phi_j(\|p_i - c_j\|)]_{i=1..N,j=1..m}$ and F be the vector $[f_i]_{i=1..N}$. The constraints points $\{p_i\}_{i=1..N}$ include both the n input points and the additional off surface points where we specify the function f value.

$$\mathbf{G} = \begin{pmatrix} \phi_1(\|p_1 - c_1\|) & \ldots & \phi_m(\|p_1 - c_m\|) \\ \vdots & \vdots & \ddots \\ \phi_1(\|p_N - c_1\|) & \ldots & \phi_m(\|p_N - c_m\|) \end{pmatrix} \tag{5.10}$$

An approximation using the least squares method implies solving the system (**??**). With the new notations, the system is

$$G^t G \cdot \alpha = G^t F. \tag{5.11}$$

The size of the matrix is $m \times m$, where m is the number of centers. The use of compactly supported functions ϕ_i leads to a sparse matrix with about 90% zero elements. In the section 7.2, different construction of the matrix are detailed.

Chapter 6

Centers

Let m be the user-defined budget of centers. The centers of the radial basis functions are selected from the vertices of the Voronoi diagram of the input points. More precisely, the centers are selected among the set of poles. As the number of poles is, typically, greater than m, a selection of m relevant poles must be performed.

We propose two strategies to sample a part of the medial axis $M(S)$: either a filtering followed by a clustering, or a greedy selection of the centers.
The filtering of the poles is based on the notion of the λ-medial axis. The filtering step allow to remove the small features or the noise and the clustering is designed to distribute the final budget of centers on the λ-medial axis with a proper sampling density.
The greedy selection consists in selecting the m poles associated with the largest maximal ball radii. For each selected pole v, we disqualify the poles which the maximal ball intersect deeply the maximal ball of v.

Our aim is to perform an adaptive sampling of $M(S)$ to obtain m points on the medial axis such that the hull of the union of inside (respectively outside) maximal balls centered on these points is an approximation of S.

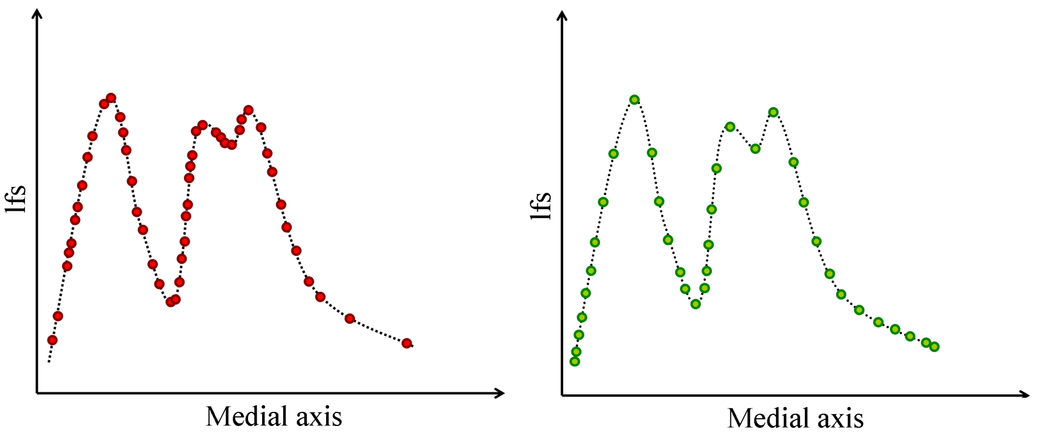


Figure 6.1: Sizing field function on the medial axis in 2D. Left (red): at each point correspond a pole. Right (green): after resampling.

The surface S is unknown, thus we do not have a continuous representation of the medial axis of S. However, we can approximate the medial axis by a subset of the Voronoi vertices (see sec.2.1.2), i.e. the set of poles approximates the medial axis. We can also approxiamate the sizing field function on the medial axis by the radius value of the maximal ball centered on the poles.

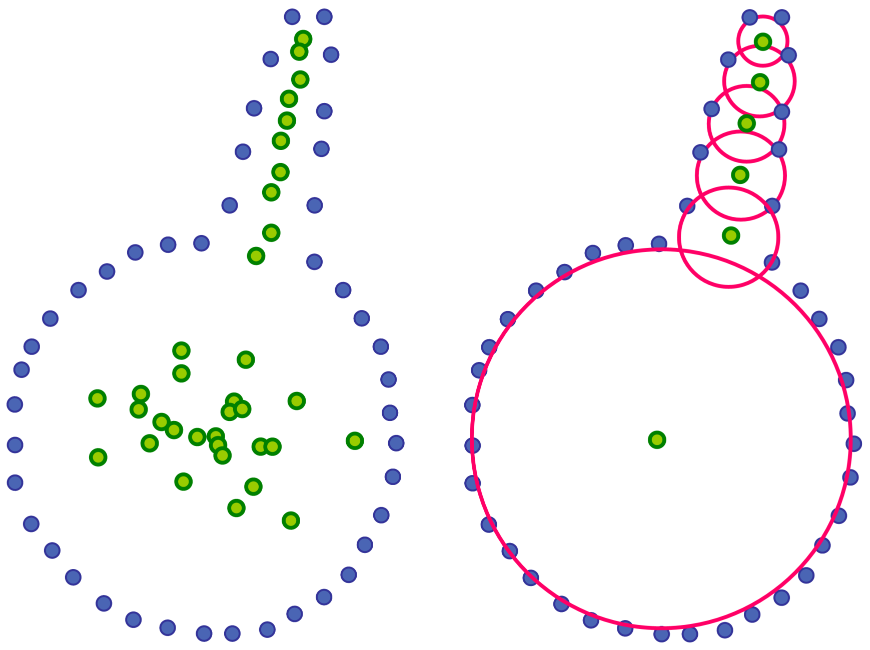


Figure 6.2: Distribution of the maximal balls. A set of blue point measured on a sphere with a bump and the set of the inside poles (green). Left: The distribution of the poles is highly dependent in the distribution of the input points Right: 6 centers (pink) are selected, their maximal ball make a good covering of the shape. The distribution of the center does not depend on the distribution of the input points.

The distribution of the poles on the approximate medial axis is highly dependent of the distribution of the input points (Fig.6.1:Left). The distribution of the maximal balls centered on the centers need to be relevant, i.e. satisfy a minimum of overlapping (Fig.6.2). Thus, a resampling of the medial axis (a *sff* discretization) is necessary to adapt the sampling density to the local geometry and the desired number of centers.

Besides the local geometry, the sampling must be adapted to m. If m is small, there is no hope to reconstruct very small details. We need to disqualify the poles which correspond to the smallest details (which are not distinguishable from noise). Indeed, small details correspond to the smallest maximal ball (see the bump on the fig.6.2). Therefore, a filtering of the poles, according to their maximal balls, had to be performed in order to adapt the sampling to the level of detail fixed by the desired number of centers.

In summary, we want to sample a part of the medial axis adapted to the level of detail and with a density independent from the distribution of the input points. We propose two strategies:

- we perform a *filtering* (sec.6.1.1) and a *clustering* (sec.6.1.2). The filtering of the poles is based on the notion of the λ-medial axis in order to remove the small features or the noise and the clustering is designed to distribute the final budget of centers on the λ-medial axis with a proper sampling density.

- we perform a *greedy selection* (sec.8.5) of m poles. Beginning by the poles with the larger radius, we then disqualify all the poles such that their support intersect deeply the selected pole. The selection stops when m poles are selected to be the centers. We obtain a centers set distribution adapted to the geometry of the shape. In addition the level of detail is adapted to the centers budget m, while poles corresponding to the smallest polar ball are not selected if m is too small.

6.1 Filtering and Clustring

6.1.1 Medial Axis Filtering

A major problem arises in the computation of the approximation of the medial axis of a sampled shape from the Voronoi vertices of the data points: the medial axis is known to be highly unstable with respect to small details of the shape. This means that even if two objects are very close for instance in term of the Hausdorff distance, they may have very different medial axis (Fig.5.2).

Thus, the set of poles extracted from the Voronoi diagram of a sampled surface is very unstable with respect to noise as well. Several approaches have been proposed to tackle this problem [AM96, DZ03, CL05].

In our work, we follow the recent work of Chazal and Lieutier [CL05], which defines the notion of λ-medial axis.

λ-Medial Axis For any point x, we denote by $\Gamma(x)$ the set points of surface S that are closest to x.

$$\Gamma(x) = \{y \in \partial\mathcal{S}, d(x,y) = d(x,S)\}\,. \tag{6.1}$$

The medial axis of S can be viewed as the set of points $x \in S$ such that $|\Gamma(x)| \geq 2$. For each point p, $\Gamma(x)$ is the smallest enclosing ball. We define the real-valued function $\gamma(x)$ as the radius of the smallest ball enclosing of $\Gamma(x)$. The λ-medial axis M_λ is defined as :

$$M_\lambda = \{x \in S | \gamma(x) > \lambda\}\,. \tag{6.2}$$

Remarks: the medial axis is equivalent to M_0 and M_λ is a closed subset of the medial axis.

Chazal and Lieutier have shown that for any value of λ which is not a singular value of the map $\lambda \longmapsto M_\lambda$, the λ-medial axis of a surface is stable under small perturbations and can be estimated from a dense sampling. Roughly speaking, restricting the λ-medial axis with increasing value of λ, smooths out both small features and noise.

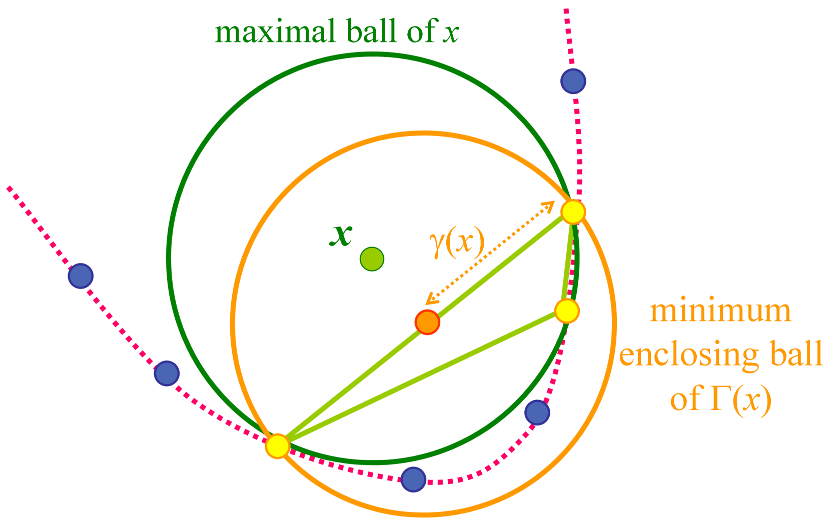


Figure 6.3: λ-medial axis criterion: the radius of the minimum enclosing ball (orange) has to be greater than λ. Relative λ-medial axis: the ratio of the radii of the two balls must be greater than $\lambda_r \in [0 \dots 1]$

Angle and Distance Removing Criterion Attali and Montanvert [?, ?] proposed a scale-invariant "removing" criterion based on a *bisector angle* and the *thickness* (Fig. 6.4). Let v be a point on the medial axis. The thickness is defined by the radius $\rho(v)$ of the

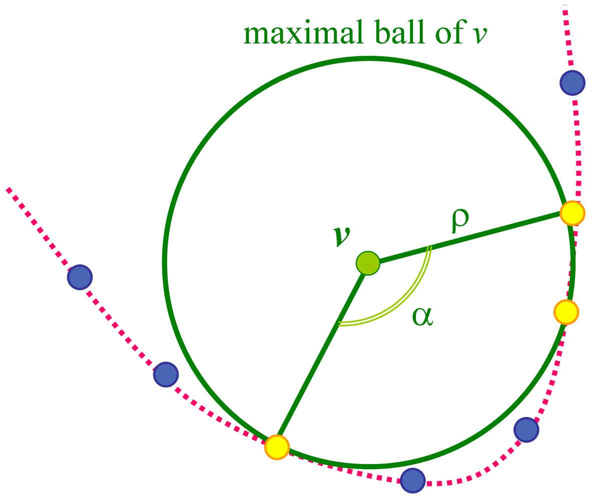


Figure 6.4: A 2D shape and Bisector angle $\alpha(v)$ and thickness $\rho(v)$

maximal ball centered at the point v. The bisector angle is defined by the maximal angle $\alpha(v)$, between v and the two of the contact points where the maximal ball centered on v touches the surface S.

Noisy vertices of the medial axis are characterized by a small bisector angle and a small

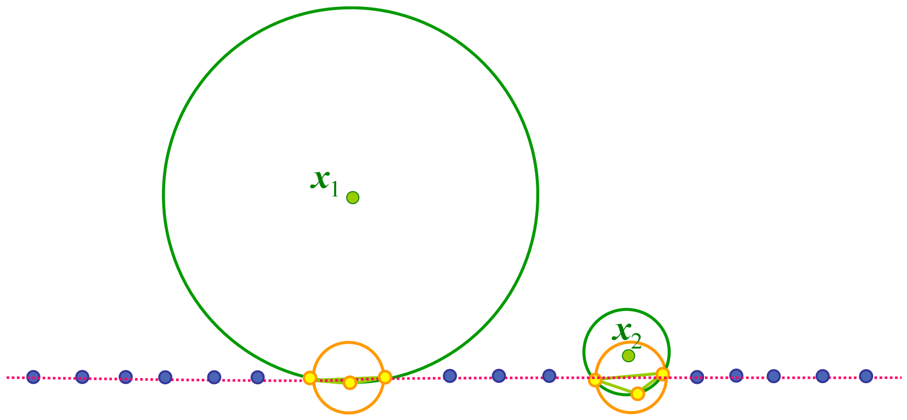


Figure 6.5: Poles to be removed by filtering.

thickness. Therefore two thresholds are proposed, α_0 and ρ_0.
The simplification consists in keeping the skeleton points v such that

$$\alpha(v) > \alpha_0 \ or \ \rho(v) > \rho_0 \tag{6.3}$$

Remarks: the relation between λ, α and ρ is given by (6.4).

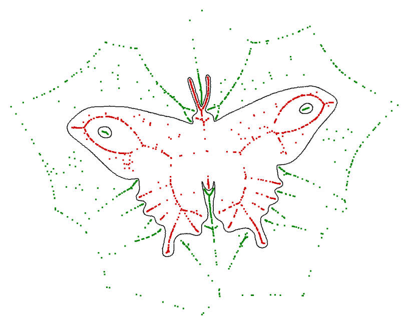
$$\gamma(v) = \rho(v) \times \sin\left(\frac{\alpha(v)}{2}\right) \tag{6.4}$$

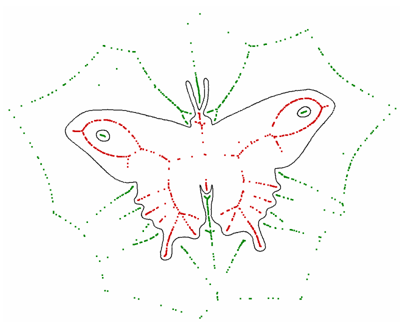
Relative λ Criterion The relation (6.4) leads to an other criterion to filter the medial axis, call the *relative* λ. The filtering is performed with the ratio between $\rho(v)$ and $\gamma(v)$. That is the ratio between the raddii of the minimum enclosing ball and of the maximal ball (Fig.6.3).

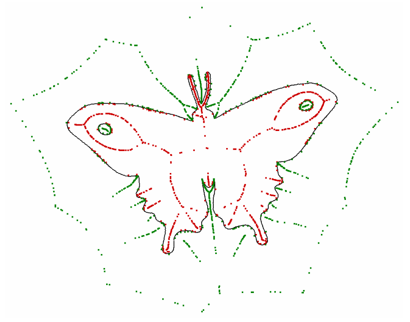
The figure 6.5 illustrates the interest of the ratio criteria. With a λ-medial axis filtering the two poles, x_1 and x_2 are removed since the radius of their minimum enclosing ball are the same, $\gamma(x_1) = \gamma(x_2)$. With the ratio criteria, the poles x_1 may be removed whereas x_2 may not removed since the ratio $\frac{\gamma(x_2)}{\rho(x_2)} >> \frac{\gamma(x_1)}{\rho(x_1)}$.

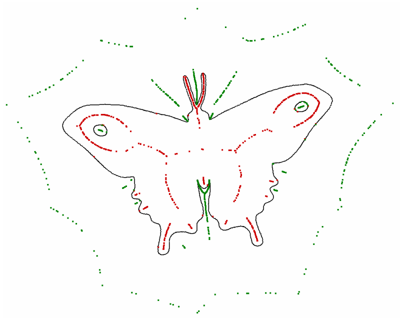
Our implementation We use the idea of λ-medial axis to perform the pole filtering in order to smooth noise and adapt the level of detail of the reconstruction to the allocated budget of centers. More precisely, this means that we determine the value λ suitable to the sampled shape and to the budget of centers, and filter out the poles which are not close to the λ-medial axis. To estimate if a pole v is close to the λ-medial axis, we compute the radius $\gamma(v)$ of the smallest ball enclosing the set $\Gamma(v)$ of sample points closest to v. Poles with radius $\gamma(v)$ smaller than λ are discarded (fig.**??** and **??**). Note that if the λ value is to large, some details are missed like the antenna of the butterfly for λ=0.01 (fig.**??**).

The figure **??** show examples of filtering We can notice that generaly the best suited filtering is to combine the two criteria (fig.6.6(e)).


(a) λ-medial axis $\lambda = 0.005$


(b) λ-medial axis $\lambda = 0.01$


(c) Relative λ-medial axis $\lambda_r = 0.1$


(d) Relative λ-medial axis $\lambda_r = 0.8$

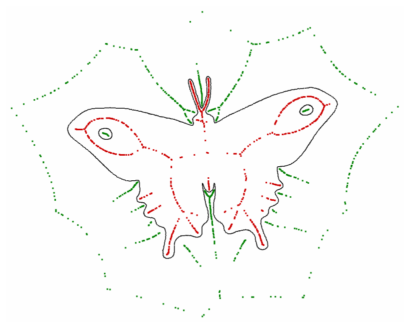

(e) Both filtering λ= 0.005 and λ_r = 0.5

Figure 6.6: Different filtered medial axis with λ criteria and ratio criteria. The figures show the poles after a medial axis filtering (red for the inside poles and green for the outside ones). To get a better sense of these parameters: the diagonal length of the bounding box of the input point set is 1.4.

6.1.2 Poles Clustering

Let m be the desired number of center. The filtered set of poles now forms a set of *possible* centers, PC. Generally, the size of PC remains larger than m. Therefore, we must select the m more relevant generator from PC, *more relevant* generator points means a set of m points which is a subsampling of PC with several properties.
Remarks: the set PC form a good approximation of the filtered medial axis of the surface S. But a sampling which is highly dependent of the distribution of the input points. In order to select m centers from PC, we perform a k-means clustering over the set of possible

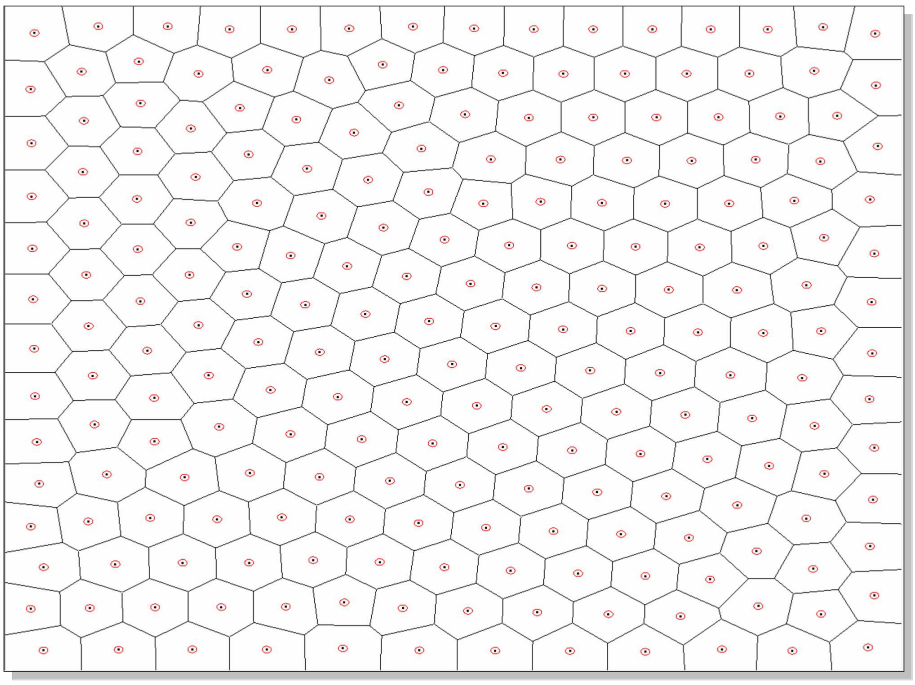


Figure 6.7: Two dimensional example: Lloyd algorithm with a uniform density. A centroidal Voronoi diagram allow a nearest neighbor partition. The red points are the Euclidean centroids.

centers [Mac67].
Remainder: Our goal is to obtain a sampling of the λ-medial axis according to the local sizing field function, *sff*. More precisely, in the center set, the distance between a center and its nearest neighbor need to be proportional to the *sff*, i.e. to the the radius of the maximal ball centered on it.

6.1.2.1 Clustering

Definition 6.1. *A k-**clustering** is a partition of a domain, Ω, into k clusters, $\{\Omega_i\}_{i=1...k}$, according to a certain density function defined on the domain.*

At each cluster Ω_i is associated a **generator** ω_i, for example the centroïd of the cluster.

To perform a clustering consist in minimizing an energy on the entire space :

$$E = \int_{\Omega_i} \int_{x \in \Omega_i} \rho(x)(x - \omega_i)^2 dx \tag{6.5}$$

where $\rho : \mathbb{R}^3 \to \mathbb{R}$ is a density function.

In order to find the minimum of this energy, two operations are performed successively until convergence (illustrated on the figure 6.8):

- Given a set of generators $\{\omega_i\}_{i=1..k}$. Optimize the cluster bound by minimizing the energy E.

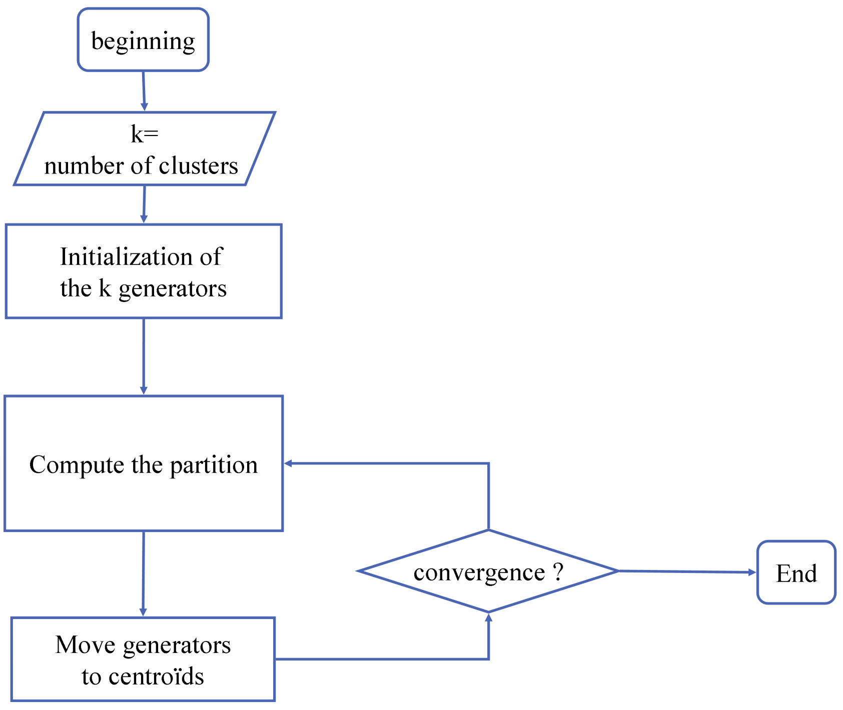


Figure 6.8: Clustering algorithm . Data : a set of n sample points and an integer k, the desired number of generators/clusters.

- Given a clusters bound $\{\Omega_i\}_{i=1..k}$. Optimize the position of the generators $\{\omega_i\}_{i=1..k}$. For a given partition, the optimal position of the generator is given by 6.6.

$$\omega_i = \frac{\int_{x\in\Omega_i} x\rho(x)dx}{\int_{x\in\Omega_i} \rho(x)dx} \tag{6.6}$$

Density Function The density function, ρ, is induced by a sizing function, μ. The sizing function defines a desired distance between a generator and its nearest neighbor (in the set of generator). More precisely, we know that the density function $\rho(x)$ must be proportional to $\frac{1}{m(x)^{d+2}}$ to obtain a cluster density matching the field μ in a underlying space of dimension d.

$$\rho(x) = \frac{1}{m(x)^{d+2}} \tag{6.7}$$

Indeed, the integrated quantity in (6.5) had to be without dimension thus ρ is $(d+2)$ dimensional, where d is the dimension of the space. Indeed, the dimension of the quadrature term dx is the dimension of the space d and $(x-c_i)^2$ is 2 dimensional.

At the convergence, all the cluster must have the same contribution int the energy (6.5), i.e. equipartitioning of the energy [DFG99].

In our work, we want a sizing function $\mu(x)$ at a given point x equals to the sizing field function $ssf(x)$ defined in the section 5.1. As the medial axis, $M(S)$, is 2 dimensional of a 3D-Surface S, we can define the density function ρ with the following equation:

$$\rho(x) = \frac{1}{ssf(x)^4}. \tag{6.8}$$

Notes that $d(x)$, the quadrature term is defined by the piece of surface corresponding to x.

6.1.2.2 The discrete case

Recalls that we want to perform clustering to sample the filtered medial axis without knowledge about the medial axis surface. However, we have a filtered sampling, PC, of an approximation of the medial axis. Thus, the clustering is performed on the sampled points, $PC = v_j$.

Let us define a clustering algorithm in the discrete case. The clustering over the filtered set PC performes a decomposition of the space into *clusters*. The set of the PC points is partitioned into clusters $\{\mathcal{C}_i\}_{i=1..k}$. Each cluster put together all the points which hold a *same feature*, i.e. all the points close according to a given similarity measure (fig.6.9).

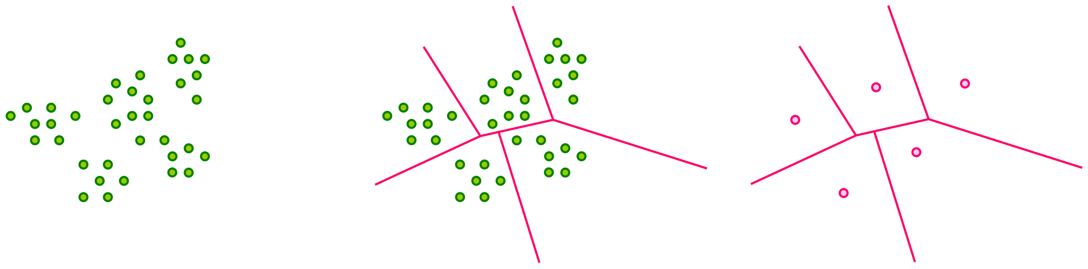

(a) Samples points (b) Space partitioning (c) Clusters and generators

Figure 6.9: 5-clustering of the green points: the space is partitioned into five clusters delimited by the pink lines. A generator is associated at each cluster.

At each cluster $\mathcal{C}_i$ is associated a **generator** c_i (Fig.6.9(c)). This generator can be the centroïd of the cell, the barycenter of the set of samples points contained in the cell or one of the point of PC.

In the discrete case, the integral (6.5) is approximated by a discrete sum (6.9):

$$E = \sum_{\mathcal{C}_i} \sum_{v_j \in \mathcal{C}_i} \rho_j (v_j - c_i)^2 d(v_j) \tag{6.9}$$

where ρ_j is a density value associated at each points v_j and $d(p_j)$ is the quadrature term defined as the surface of medial axis corresponding to the p_j.

In order to find the minimum of this energy, like the continuous case, two minimizations are performed successively until convergence:

- Given a set of generators $\{c_i\}_{i=1..k}$. The clusters $\{\mathcal{C}_i\}_{i=1..k}$ are computed to minimize the energy E.

- Given a set of clusters $\{\mathcal{C}_i\}_{i=1..k}$. The generators $\{c_i\}_{i=1..k}$ are computed such that they minimize E using the equation 6.10.

$$c_i = \frac{\sum_{v_j \in \mathcal{C}_i} v_j \rho_j d(v_j)}{\sum_{v_j \in \mathcal{C}_i} \rho_j d(v_j)} \tag{6.10}$$

Local Density Function The density value ρ_j associated to the point $v_j \in PC$ is defined like the ρ function in the continuous case (6.8). As the discrete value of the $ssf(v_j)$ is r_j, the radius of the polar ball centered on v_j, we obtain the following equation:

$$\rho_j = \frac{1}{r(v_j)^{d+2}} \tag{6.11}$$

In our case $d = 2$, because the filtered poles approximate the medial axis, which is a generically a two-dimensional manifold.

$$\rho_j = \frac{1}{r(v_j)^4} \tag{6.12}$$

Quadrature Term The quadrature term expresses the poles distribution. More precisely, given a pole v_i, the term $d(v_i)$ takes into account the local density of the poles in the neighborhood of v_i.

A space quadrature of the space is performed in order to associate at each pole v_i its contribution. We want to compute the part of the medial axis which correspond to v_i (fig.6.10:Left).

Let v_i be a pole. We compute the area of the intersection of the filtered medial axis and the Voronoi cell of v_i in the Voronoi diagram of the poles.

Most of the Voronoi cell shapes like a pencil (oblong) (fig.6.10). Therefore, the cell volume can be approximated by Volume(filtered medial axis$\bigcap$cell(v_i)) $* r(v_i)$. This approximation allowes us to compute an approximation of the area $d(v_i)$ with $\frac{V(v_i)}{r(v_i)}$, where $V(v_i)$ is the volume of the Voronoi cell of v_i in the Voronoi diagram of the (filtering) poles and $r(v_i)$ is the radius of the maximal ball.

6.1.2.3 Experiments

Distance between two poles Two poles may be close but their respective radius can be very different. Therefore, these two poles may not be in the same cluster. We chose to compute the distance between two poles not in 3D but in 4D. More precisely, we add the radius as the fourth poles coordinates:

$$d((v_i, r_i), (v_j, r_j))^2 = \|v_i - v_j\|^2 - (r_i - r_j)^2. \tag{6.13}$$

Initialization Problem The clustering algorithm is strongly dependent on the choice of the k generators initializations. To overcome this problem, the initialization must take

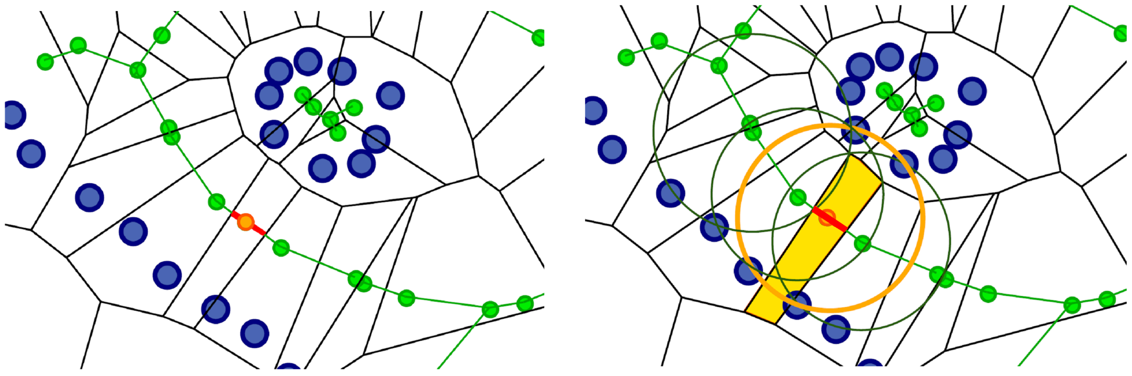


Figure 6.10: Quadrature term approximation (detail of the butterfly wing). Left: Voronoi diagram of the poles (green), we want to compute quadrature term (red segment size) for the red point. Right: The volume of the orange Voronoi cell can be approximated by the product of the surface of the red segment and the radius of the green maximal ball.

into account the desired density. We are going to perform several clustering successively such that, at each iteration:

- the generator number k increase,
- the initial generator set is constructed according to the final generator set of the previous iteration.

At the end of the iteration l, the generator set is $R^l = \{c_i^l\}_{i=1..k}$, i.e. a set $C^l = \{\mathcal{C}_i^l\}_{i=1..k}$ of clusters. For the next iteration $l+1$, the set of generators R^{l+1} is initialized with R^l and a set of $\frac{k}{2}$ new generators. This $\frac{k}{2}$ points are located in the area where there is a deficit of density. We evaluate the local density at each c_i by computing the ratio between the distance from c_i to its nearest neighbor in R^l and $\mu(c_i)$. We add a new generator for each $\frac{k}{2}$ generators which own the largest ratio.

The algorithm stops when the number of clusters is equal to m.

Non Uniform Clustering vs Uniform Clustering In this paragraph, we want to justify the use of a non uniform clustering.

The *uniform clustering* is performed using a uniform density function, i.e. $\rho(x) = 1\ \forall x \in \Omega$.

In the figure 6.12, we can compare the resulting center set after a uniform clustering (left) and a non uniform clustering (right). The model used for is a ball with two bump (Fig.6.11), the two bump are subsampled while the ball is oversampled. When a uniform clustering is performed, the generator distribution depends on the sample distribution. That is the centers distribution is related to the sampling density of the data points and, therefore, we obtain not enough centers in the bump area (fig.6.12 right). In contrast, our non uniform clustering allow us to better distribute the centers: the part of the medial axis where the local sizing field is small are more finely sampled (fig.6.12 left). The figure 6.13, by drawing the inside maximal ball, shows more clearly the effect of the two clusterings.

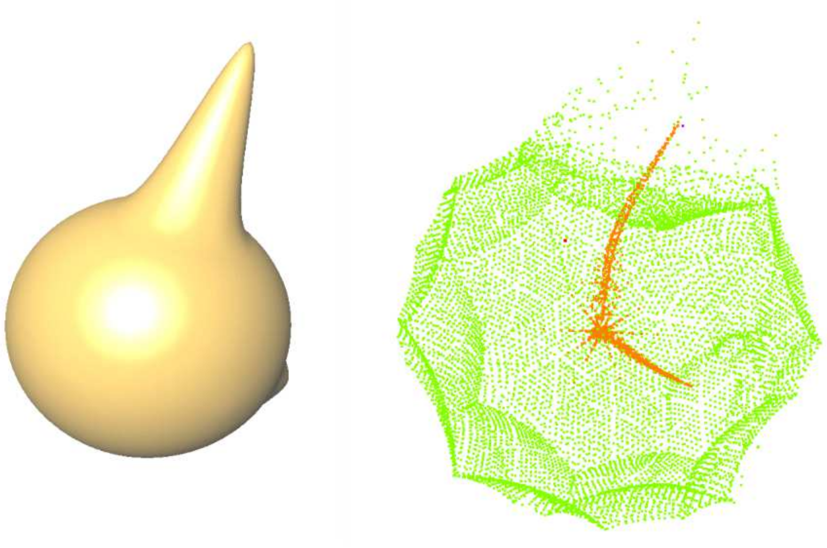


Figure 6.11: Left : a ball with 2 bumps. Right : all the extracted poles (oranges for the inside poles and green for the outside ones).

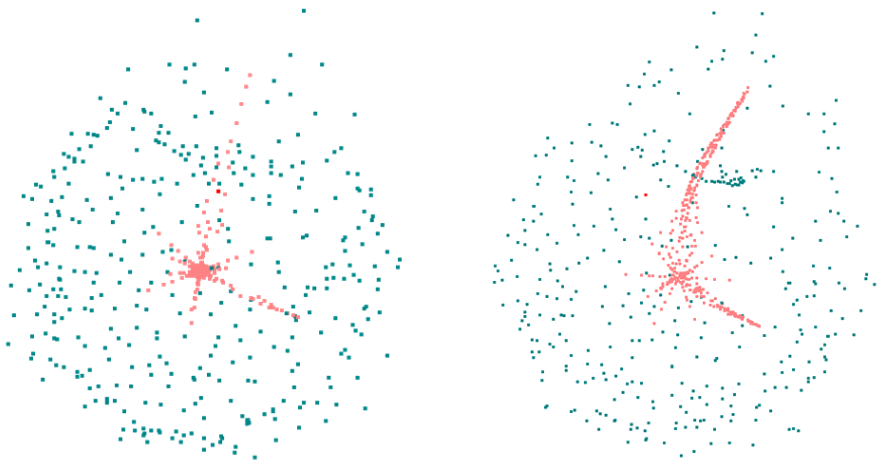


Figure 6.12: Left: the centers after a uniform clustering. Right: the centers after a non uniform clustering. (red for the inside centers and green for the outside ones)

After convergence of the clustering procedure, the centroid of each cluster is replaced by the closest pole within its cluster, so that the final centers are guaranteed to be located near the medial axis of the sampled surface.

6.2 Greedy Selection

Recalls, let m be the desired number of centers and a PC the set of *possible centers*, i.e. the set of poles. We want to obtain a center set such that the center points sample a part of the medial axis of S. The center set density had to be independent from the distribution of the centers. The choice of centers must be adapted to the level of detail given by m

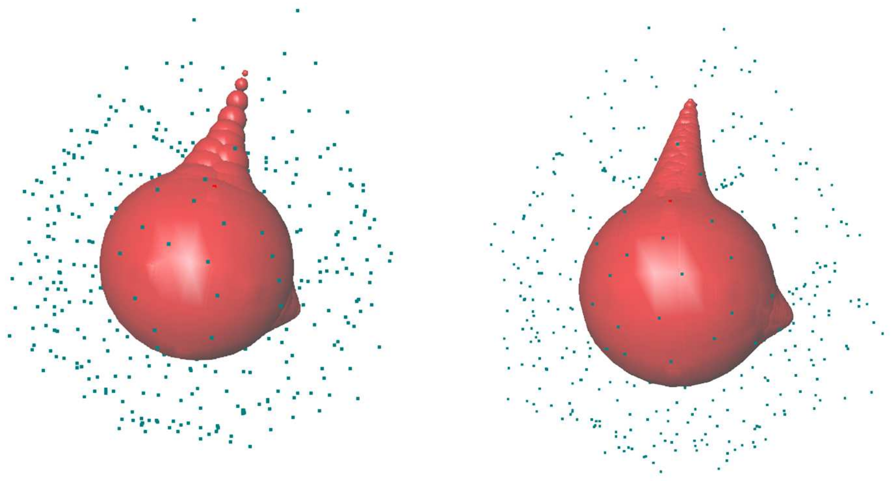


Figure 6.13: Left: centers after a uniform clustering. Right: centers after a non uniform clustering. (red for the inside maximal ball and green for the outside centers)

(fig.6.1:Right).

The main idea is to perform a greedy selection of the centers according to the sizing field function, *sff*. In other word, the distance between a center c and its nearest neighbor in the center set must be proportional to $ssf(c)$. The proportionality coefficient depends on a user defined maximum overlapping rate ρ between the polar ball of two selected centers.

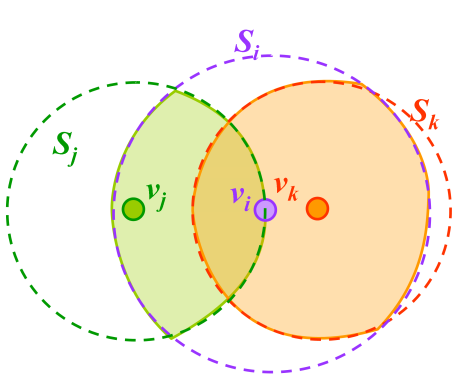
We do not know the exact medial axis surface, i.e. the sizing field function, however we know a discrete approximation of the medial axis as the pole set, PC. For each pole v, the sizing field function is approximated by the radius of the polar ball centered at v, i.e. $ssf(v) = r_v$.

Our idea is to perform a greedy algorithm such that relevant poles are selected and the redundant ones are disqualified. The density of the center points must be conversely proportional to the sizing field function, i.e the bigger the radii the fewer poles we want (Fig.6.2). In order to obtain a relevant density, when a pole is selected to be a center, several poles are disqualified.
The poles are added iteratively in the set of centers until the center set size reaches m, beginning by the ones associated to the larger radius. Let v_i be a selected pole. The closest pole to v_i may be disqualified, that is all the poles contained v_j in the v_i-polar ball are candidate for the disqualification.

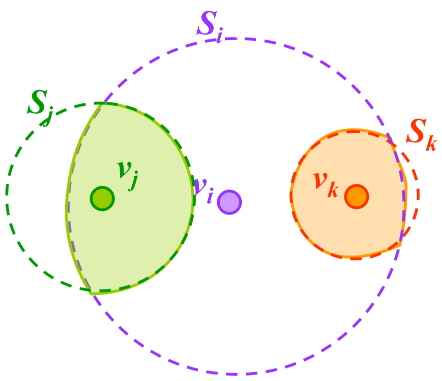
Our approach is to check if the polar ball intercept deeply the v_i-polar ball are disqualified. In order to test the deeply intersection of two polar ball, we compare the volume of the intersection of the two polar balls $Vol(S_i \cap S_j)$ with volume of the smallest polar ball $Vol(S_j)$ (Fig.6.15).

Notes that, as illustrates the figure 6.14, either the distance from the pole v_j and v_i (Fig.**??**) or the difference between the radii of the corresponding polar balls (Fig.**??**) are not discriminant.

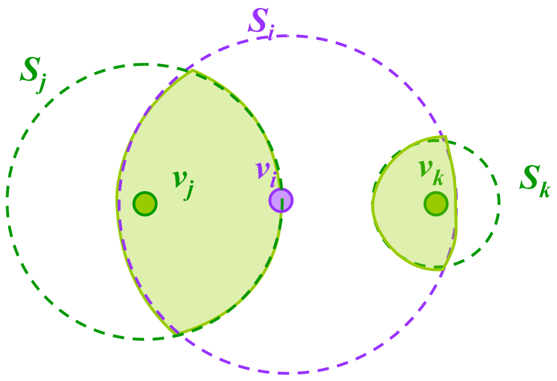
An user defined overlapping rate threshold ρ allows to decide if a pole is disqualified or



(a) The polar ball radius is not discriminant



(b) The distance to v_i is not discriminant



(c) The 2 poles are not disqualified but the pole v_k is selected only if the the budget of center is sufficient.

Figure 6.14: Greedy selection. Different cases of polar ball intersections. Let v_i be a selected pole. According to the intersection between the polar ball the poles v_j and v_k are disqualified (red ones) or not (green one).

not. When the number of selected centers reach m, several poles non disqualified and non selected may remain. These poles are associated to a too small radius for the user defined number of centers.

Therefore, the center set is adapted to the desired density and to the level of detail given by the center budget (Fig.**??**).

Algorithm First, we sort the poles according to its radii. Then, select the pole with the larger radius and disqualify the poles which maximal sphere intersect deeply the maximal sphere of the selected pole. More precisely, let p be the selected pole and S be the maximal ball of S_p. For each pole v in S_p, if the volume of the intersection of S_p and S_v, the v maximal sphere, is greater than $\rho \times Vol(S_v)$ the pole v is disqualified; where ρ is a parameter : the threshold of the overlapping rate. That is the set of disqualified D_p is

$$D_p = \{v \in S_p | Vol(S_v \cap S_p) > \rho \times Vol(S_v)\} \tag{6.14}$$

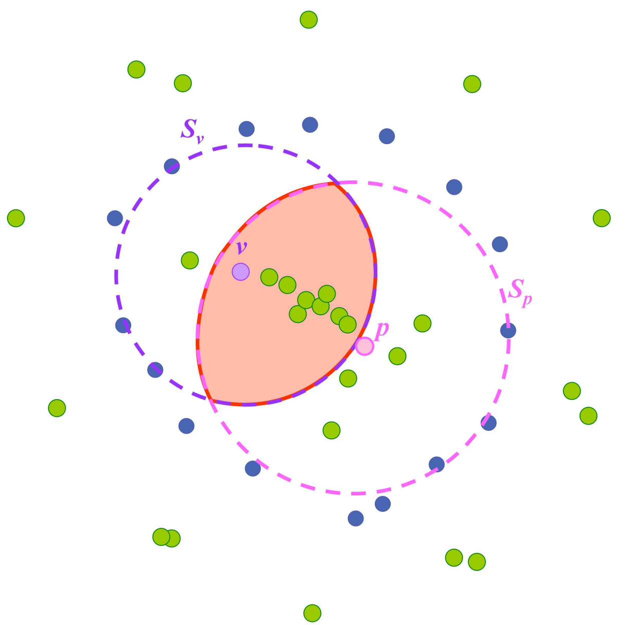
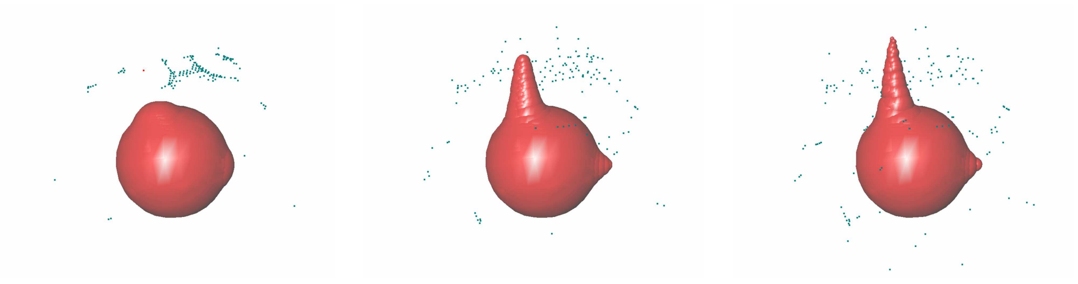


Figure 6.15: Given a selected poles p and a pole v, the volume of the red area is compared to the volume of S_v.

As it shown on the figure 6.16, the value of ρ determine the level of details. a trade between the level of detail and the overlaping of the balls may be found. Notes when $\rho = 1$ (Fig.6.16(a)) the m selected centers are the m poles associated to the largest polar ball.



(a) $\rho = 1$ (b) $\rho = 0.8$ (c) $\rho = 0.5$

Figure 6.16: Examples of greedy selection for a bumped sphere (10K input points). 300 centers are selected using the greedy selection (inside centers and their polar balls in red and the outside centers in green).

Chapter 7

Implementation

Given a set of input points $P = \{p_i\}_{i=1..n} \subset \mathbb{R}^3$ measured on a surface S, we want to construct a surface S' that approximates S.
We recall that we restrict to the case where the surfaces divide the space into two subspace: a bounded volume tagged as inside and an unbounded volume tagged as outside.

The reconstructed surface S' is defined as

$$S' = \left\{x \in \mathbb{R}^3 : f(x) = 0\right\} \tag{7.1}$$

where $f(x)$ is expressed as a weighted sum of basis function ϕ centered on a set of center points c_j (7.2).

$$f(x) = \sum_{j=0}^{n} \alpha_j \phi(\|x - c_j\|) \tag{7.2}$$

The function solution, i.e. the vector of coefficients α, is computed by minimizing a least squares error (7.3):

$$f^* = \arg\min_{f \in \mathcal{F}} E(f). \tag{7.3}$$

Given a set of N constraints $\{x_i\}_{i=1..N}$ where the function f is known. The minimization consist in solving the following linear system:

$$G^t G \cdot \alpha = G^t F. \tag{7.4}$$

where $G = [\phi(\|x_i - c_j\|)]_{i=1...N, j=1...m}$.
Our algorithm proceeds as follows:

1. Compute the Delaunay triangulation of the input points;
2. Extract the set of poles from the Voronoi vertices;
3. Classify the poles as inside or outside;
4. Select the centers points from the set of poles;
5. Assemble the matrix G;

6. Solve the system.

We are going to detail the implementation of the step 3, the poles classification algorithm, and 5, the construction of the matrix in an efficient way.

7.1 Poles Classification

Recall the set of **poles** is a subset of Voronoi vertices.
Given a point $p \in P$, let $V_p = V_{P,p}$ be its bounded Voronoi cell in the Voronoi diagram of the point of P. Two poles are extracted for each bounded cell V_p (see sec.2.1.2). The first pole v_1 is the Voronoi vertex in V_p with the largest distance to the sample point p. The second pole is the Voronoi vertex v_2 in V_p furthest away from p in the opposite half space of v_1.
Note that if the Voronoi cell V_p is unbounded p own a unique pole, Voronoi vertex the furthest away fron p in V_p.

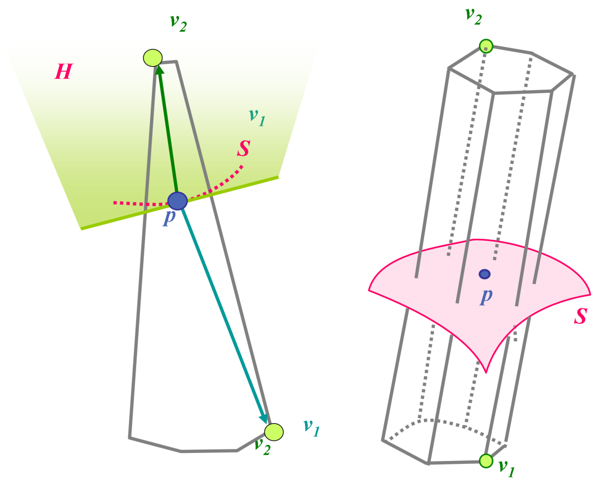


Figure 7.1: Voronoi cell in 2D and 3D

If the sampling is dense enough, the vectors $\vec{pv_i}$ are a good approximation of the normal to the surface at the point p (Fig.7.1). Thus, if the sampling is dense enough v_1 and v_2 lie on each side of the surface.

As we use poles as additional constraints, poles v_i need to be labeled inside or outside in order to affect a sign to the value $f_i = f(v_i)$.

To avoid deal with infinite Voronoi cells, a set of bounding points, B, lying on an augmented of the P convex hull, are added to the input point set P (Fig.7.2). Therefore, the all Voronoi cell of the input points are bounded, i.e. at each point $p \in P$ correspond two poles.

A pole graph (Fig.7.3) may be is constructed. The graph nodes represent the poles and there is an edge joining two poles v_i and v_j if:

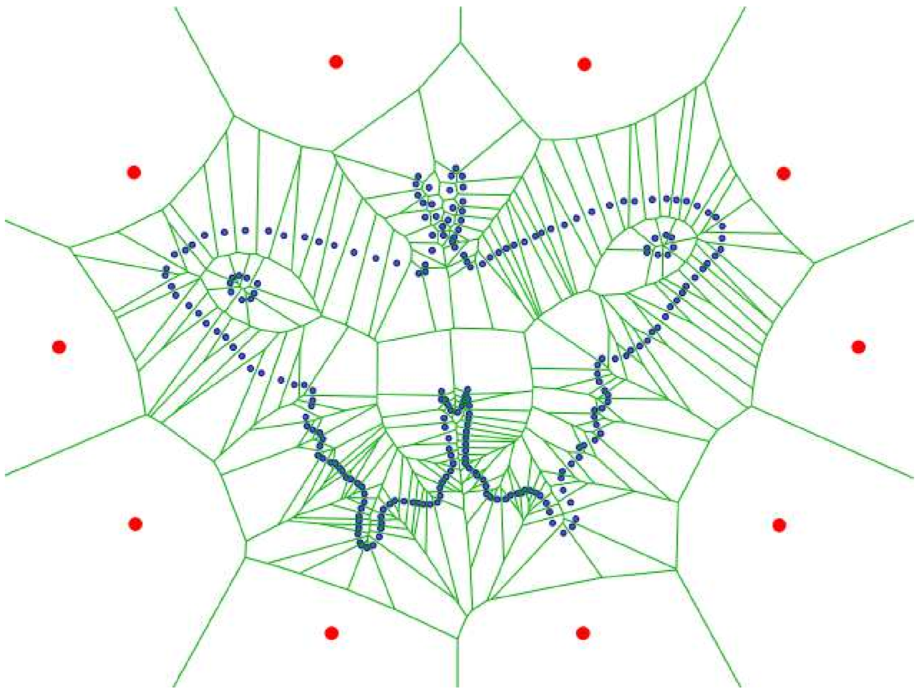


Figure 7.2: The Voronoi diagram of the input points set augmented by the bounding box points (red points)

1. v_i and v_j are the two poles of a given Voronoi cell V_p,
2. v_i and v_j are neighbor in the Voronoi diagram of the input points, P. That is the dual Delaunay tetrahedra associated to v_i and v_j are adjacent in the Delaunay triangulation of the input points.

Note that in the case 1, the probability of the poles to be on opposite side of the surface is hight. Conversely, in the case 2 the poles v_i and v_j can be either in the same side or not but it is most probable that the poles are lying on the same side (Fig.7.3).

The main idea of the labeling algorithms is to associate at each poles a probability measure on a temporary label. More precisely, the labels and their probability are propagated in the poles graph. The points of B are used to initialize the algorithm.

In [ACK01] (see sec.2.1.2), the labels are propagated using the two manners described above. The algorithm used a priority queue containing the poles with a non definitive label. The priority is based on the label probabilities.

In order to initialize the algorithm, the poles of the bounding points are labeled as outer with a probability equal to one. Then the pair (label, probability) is propagated utile all the poles are labeled. The pole with the highest probability get a definitive label and is removed from the priority queue.

Let v and u be two poles, $\phi(v, u)$ is defined as the angle between the polar ball of u and v (Fig. 7.4). The labeling algorithm is the following :
Let v_1 be a pole with a definitive label.

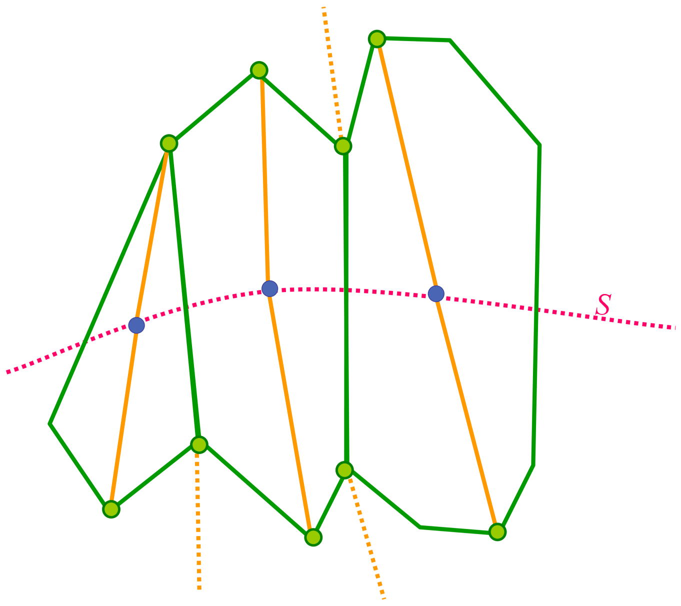
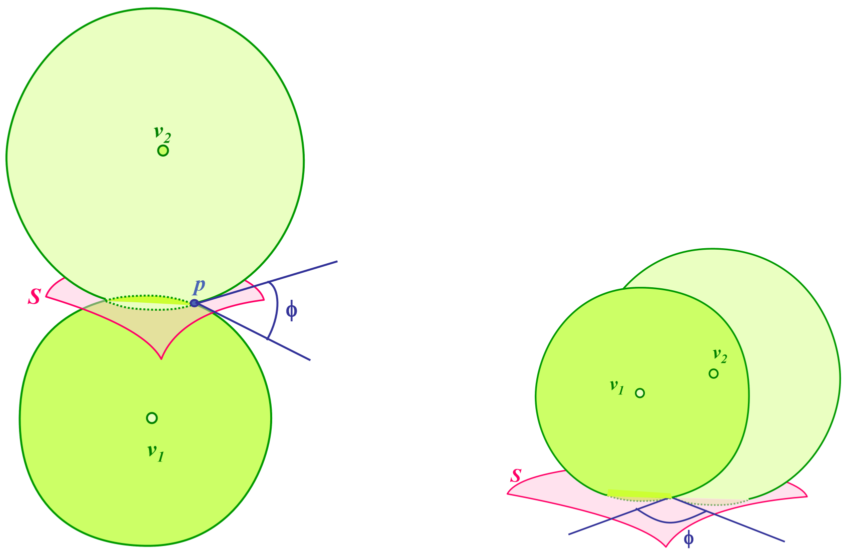


Figure 7.3: Edges of the poles graph in 2D. There are two king of edges joining two poles. Orange edges when the 2 poles share a vertex (case 1) and Green edges, Voronoi edges, when the poles are neighboor in the Voronoi diagram (case 2)

- In the case 1: the temporary label of v_2 is fixed to the opposite v_1 label with a probability computed from the cosines of ϕ (Fig. 7.4(a)).
- In the case 2: the temporary label of v_2 is fixed to the v_1 label with a probability proportional to the cosines of ϕ (Fig. 7.4(b)).



(a) two maximal balls shallowly intersect

(b) two maximal balls deeply intersect

Figure 7.4: The two cases of maximal ball intersection. Left: when the two poles are on each side of the surface. Right: when the two poles are on the same side of the surface.

This classification method assumes some conditions over the sampling : ϵ-sampling (see appendix .6) . Recalls a sample is an ϵ-sampling when the distance from any surface point x to the nearest sample point is at most a small constant ϵ times the distance to the medial axis.
As the point set is scattered on the surface, we can not guarantee such sampling conditions, and therefore do not use the original version of this algorithm. The selection of outer versus inner labels can fail with undersampling, noise, or lack of smoothness of the surface S. Some heuristics which characterize the "shape" of Voronoi cells used have been introduced to discard poles. If the cells are not long and skinny, they are rejected.

Algorithm:

```
Algorithm PolesClassification
Input: The Voronoi Diagram of the input points $P$
Output: A set of labeled poles
1.   for all poles $v$
2.       do initialize $in(v) = out(v) = 0$
3.           insert p in the priority queue $\mathcal{Q}$for each pole $v$ adjacent to points of $B$
4.        do out(v) = 1
5.            Update Priority(v) in $\mathcal{Q}$
6.     while Q is not empty
7.        do Remove the top element $v$ of $\mathcal{Q}$
8.            if $in(v) > out(v)$
9.               then $label(v) = in$ and $tmp(v) = in(v)$
10.              else  $label(v) = out$ and $tmp(v) = out(v)$
11.           for each input point $p$ of which $v$ is the pole
12.                do let $u$ be the other pole of $p$
13.                    $opp(label(v))(u) = max(tmp(v) * w_{uv}; opp(label(v))(u))$
(* $opp(in) = out$, $opp(out) = in$, $w_{pq} = cos(\phi)$ (Fig.7.4(b)) *)
14.                    Update Priority(u)
15.            for each deeply intersecting neighboring poles u
16.                 do $(label(v))(u) = max(tmp(v) * w_{pq}; (label(v))(u))$
(* $w_{pq} = cos(\phi)$ (Fig.7.4(b)) *)
17.                     Update Priority(u) in $\mathcal{Q}$
```

In the Eigen crust [KSO04] (see sec.2.1.6), the classification algorithm consists in a normalized cut in the pole graph described above. The eigen vector corresponding to the smallest eigen value determines a division of the graph into two subgraph containing inside and outside poles.

In our algorithm, we perform a variant, due to F.Cazals (internal communication), more efficient and more robust against to noise. In the two approaches presented above, only the poles are took into account. Our approach is to label the poles and at the same time to orient the normals at the input points. We use a priority queue, $\mathcal{Q}$, in which the nodes are either poles or vertices of Delaunay triangulation. More precisely are either a pole with a temporary label and a probability or a point with an temporary oriented normal and a a probability.

7.2 Computation of the Matrix

Given a set of N constraints, $\{x_i\}_{i=1...N}$ associated with N scalar values $F = f_{i i=1...N}$, a set of m centers, $\{c_j\}_{j=1...m}$ with an associated radius r_j, and a class of basis functions, ϕ. Our problem is to find a function f minimizing the energy functional (7.3). The minimization consist in solving linear system $G\alpha = F$, where G is given by (7.6), in the least squares sens, i.e. solving the system (7.5).

$$G^tG \cdot \alpha = G^tF. \tag{7.5}$$

where G is given by:

$$\mathbf{G} = \begin{pmatrix} \phi_1(\|x_1 - c_1\|) & \dots & \phi_m(\|x_1 - c_m\|) \\ \vdots & \vdots & \ddots \\ \phi_1(\|x_N - c_1\|) & \dots & \phi_m(\|x_N - c_m\|) \end{pmatrix} \tag{7.6}$$

A challenge is to compute efficiency the matrix G or G^tG.

Notations: In the following, given a center c_j, $\phi(\|x - c_j\|) = \phi_j(x)$. Recalls ϕ_j is a compactly supported function defined as (7.7):

$$\phi_j(x) = (1 - \frac{\|x - c_j\|}{\|S_j\|})^4_+(1 + 4\frac{\|x - c_j\|}{\|S_j\|}), \tag{7.7}$$

where the symbol $+$ means $(x)_+ = x$ if $x > 0$ and $(x)_+ = 0$ otherwise. S_j is the supporting ball of ϕ_j, centered on c_j, that is $\phi_j(x) = 0 \ \forall x \notin S_j$. The size of the support, $\|S_j\|$, is a constant time the radius of the polar ball centered in c_j.

$$\|S_j\| = c^{st} * r_j \tag{7.8}$$

7.2.1 Trivial Method

The trivial method constructs the matrix G (7.6) by computing $\phi_j(x_i)$ for all center c_j and all constraints x_i. Then, we can compute the matrix $A = G^tG = [a_{i,j}]$, $i, j = 1..m$

$$a_{i,j} = \sum_{k=1}^{N} \phi_i(x_k)\phi_j(x_k) \tag{7.9}$$

7.2.2 Selective Method

As the radial basis function ϕ_j are compactly supported, the matrix G and also the matrix G^tG are sparse.
In practice, only about ten percent of matrix coefficients are not zero. Therefore, it would be interesting to compute only non zero coefficients.We can note that $a_{i,j}$ (7.9) is zero when the supports S_i and S_j do not intersect (fig.7.5)

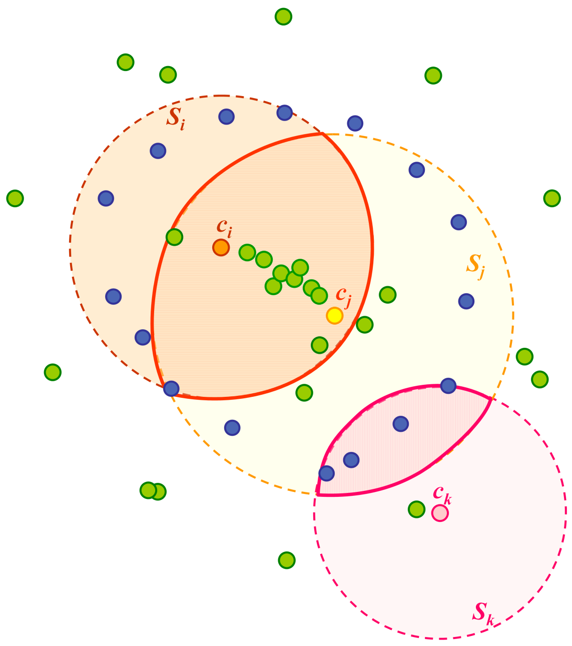


Figure 7.5: Input points (blue) and centers (green). The support of ϕ_i and ϕ_j (S_i and S_j) intersect and share several centers so $a_{i,j} \neq 0$. The support of ϕ_i and ϕ_k (S_i and S_k) does not intersect so $a_{i,k} = 0$.

$$a_{i,j} = \begin{cases} 0 & S_i \cap S_j = \emptyset \\ \sum_{x \in S_i \cap S_j} \phi_i(x)\phi_j(x) & \text{else} \end{cases} \quad \forall x \in X \tag{7.10}$$

Our idea is to go through the centers and seek for the constraints contained in the support of two centers.

Data Structures :

For each center c_j, the constraints x contained in S_j are listed. In order to seek for the constraints in S_j we construct a *kdtree* over the constraints. This kdtree allows us to not visit all constraints. The constraints are visited according to their increasing distance to c_i and the algorithm stop when the current constraints x_i is such that $\|x_i - c_j\| > \|S_j\|$.

Note that the values $\phi_i(x)$ are precomputed for all $x \in S_i$ during the computation of the diagonal term. The centers are sorted according to their support and the centers with the smallest support are processed first.

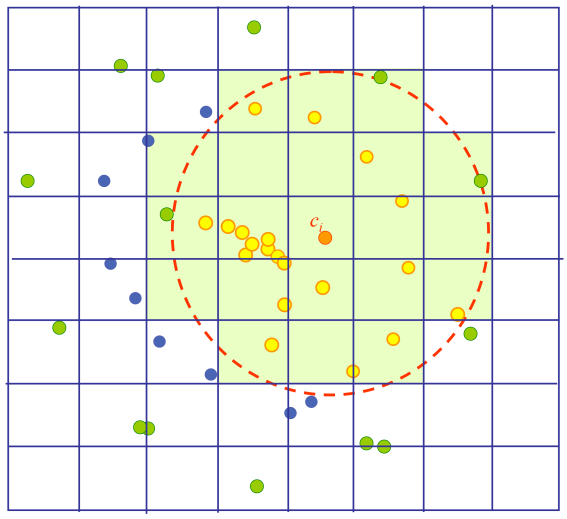


Figure 7.6: Given a center c_i, a list of constraints (yellow points) is constructed using kdtree. Only the constraints in the green area are visited.

Algorithm : The algorithm *MatrixConstruction* describes the matrix construction:

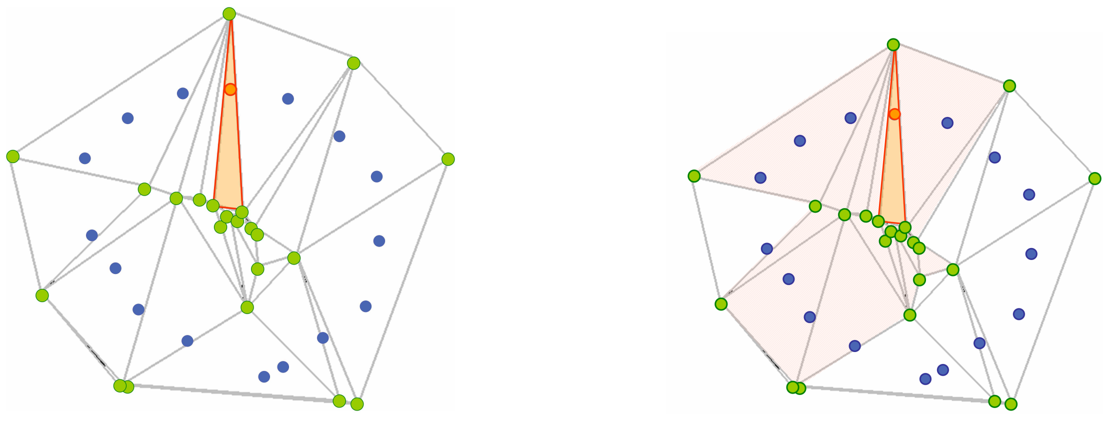
Algorithm *MatrixConstruction*

```
(* Construct the matrix $A = G^tG$ and the vector $b = G^tF$. *)
1.  for all $c_i$
2.          Compute the $i^{th}$ diagonal term $a_{i,i}$ of the matrix A
3.          Compute the $i^{th}$ coordinate of the vector $B = (G^tf)_i$.
4.          All constraints belonging to the support $S_i$ are put in a list $\mathcal{L}_i$.
5.          for all $c_j$ such that $i < j < m$
6.              do $A[i][j] = 0$
7.                  if $S_i \cap S_j \neq \emptyset$
8.                    then for all constraints $x$ in $\mathcal{L}_i$
9.                              if $x \in S_j$
10.                               then $A[i][j] += \phi_i(x) \times \phi_j(x)$
11.         $A[j][i] = A[i][j]$
12.
```

7.2.3 Dual Method

In the previous algorithm, we perform a double loop on the centers. Given a centers c_i with a large support S_i which contain n constraints (with n sufficient large). Suppose that S_i intersect slightly S_j, the support of c_j, about s constraints in common ($s < n$). In order to compute $a_{i,j}$, the n constraints contained in S_i are tested to be in S_j, i.e. n tests while only s terms in the sum (7.9).

Therefore, it may be more interesting to perform the dual method. The idea now is to go thought the constraints and seek for the centers which contain x in their support. In the following, we call this centers *x-relevant centers*.



(a) the point is located in DTC. (b) Propagation in the centers triangulation.

Figure 7.7: Dual method. Input points (red), the centers (black) and the Delaunay triangulation of centers (gray).

The *dual* method consists in a loop over the constraints, for each constraint a list of *x-relevant centers* is constructed. A center is tagged *x-relevant* when its support contains the current constraint x (fig.7.7(a)).

To avoid a greedy algorithm which test all centers for each constraints, a Delaunay triangulation of centers ($\mathcal{D}_C$) is implemented. The "current" constraint, x, is located in a cell $\mathcal{C}$ of the $\mathcal{D}_C$. The cell $\mathcal{C}$ is used as a base for traversal algorithm in order to collect all centers which contain the currant constraint in their support (fig.7.7(b)).

A stack of candidates cells is constructed to collect the *x-relevant centers*. A candidate cell satisfying this two conditions

- at least one on its vertices is *x-relevant centers*
- at least one on its vertices have not been visited yet for the current constraint x

The propagation in the centers triangulation, $\mathcal{D}_C$, stop when there is no candidates cells anymore. This strategy suppose that the *x-relevant* centers for a given constraint is connexe.

Data Structures :

The Delaunay triangulation of the centers, $\mathcal{D}_C$, allows to structure the center set with neighborhood relation. Given a constraints x, $\mathcal{D}_C$ allows to collect the centers close to x by a traversal algorithm.

The localization in $\mathcal{D}_C$ may be expensive if the search start with an arbitrary cell of the triangulation. We, thus, initialize the search with the cell of the previous current constraint. In order to obtain a good initialization, the constraints are structured in a **space filling curve**. A space filing curve is a line passing through every point in a space, in a particular order, according to some algorithm. Note that curves pass through points once only (fig.7.8(a)).

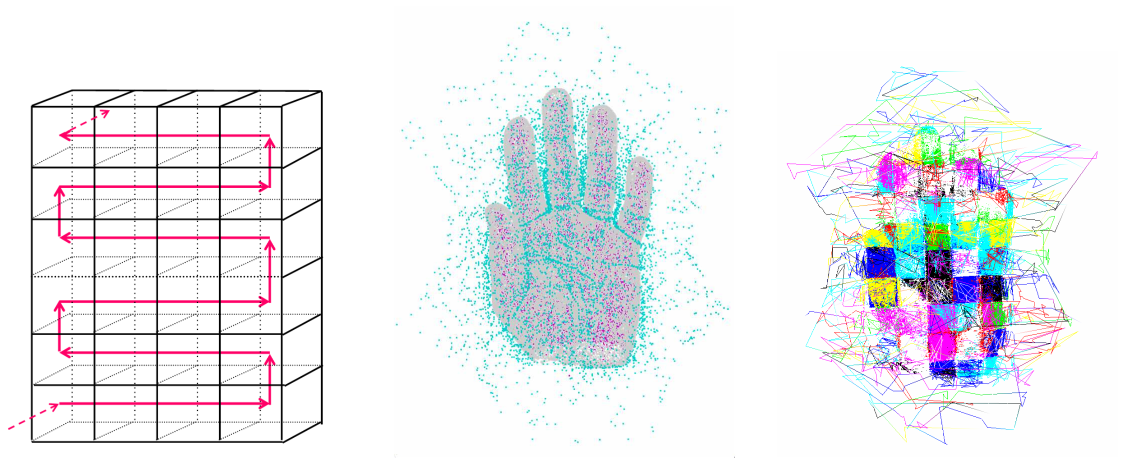

(a) Space filling curve in 3D (b) Set of constraints points (c) Space filling curve on the constraints

Figure 7.8: Space filling curve is a "continuous curve" in 3-dimensional space.

Algorithm : The algorithm *MatrixConstruction2* performs a loop over the constraints $\{x_i\}$ in order to find for each of then all the centers which their support contains it.

Algorithm *MatrixConstruction2*
Input: X a set of constraints structured as a space filling curve *sfc* and $\mathcal{D}_C$ a triangulation of the centers.
Output: The matrix $A = G^tG$ and the vector $b = G^tF$ with the 2^nd method.

```
1.  Initialization to zero of $A[i][j]$ and $b[i]$,$i, j = 1..m$.
2.  for all sfc box $sfc_i$
3.      do for all constraints $x \in sfc_i$
4.              do Locate $x$ in $\mathcal{D}_C$.
5.                 Let $\mathcal{C}$ the cell which contains $x$.
(∗ Seek after the x-relevant centers ∗)
6.                 push $\mathcal{C}$ is a cellStack
7.                 while cellStack non empty
8.                     do pop the first cell $\mathcal{C}_k$ in cellStack
9.                         for all neighbor tetrahedra $T$ of $\mathcal{C}_k$
10.                            do for all vertex $c_i$ of $\mathcal{C}_{||}$
11.                                do if $c_i$ have not been seen yet for $x$
12.                                    then Tag $c_i$ as seen
13.                                        if $\|x - c_i\| < \|S_i\|$
14.                                          then Tag $c_i$ as x-relevant
15.                                               add x in x-relevant centers list
```

```
16.                         compute and store $\phi_i(x)$
17.                 if at least one of vertices is x-relevant
18.                     then add $\mathcal{T}$ in cellStack
19.
20.         for all center pairs $(c_i, c_j)$ in x-relevant centers list
21.             do Update the values $A[i][j]$
22.
```

Chapter 8

Results

We have implemented our algorithm in C++. The Voronoi diagram and Delaunay triangulation are computed using the CGAL library [FGK+00]. The linear system is solved using the TAUCS library [Tol01]. We use a Delaunay-based surface mesher elaborated by Rineau et al [?].

8.1 Fitting Accuracy

To evaluate the fitting accuracy, we use the *Taubin distance* [Tau94] from the input points (8.1)

$$Err(f) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{f_i - f(p_i)}{\|\nabla f(p_i\|}\right)^2. \tag{8.1}$$

This error is a sum of first order approximation of the Euclidean distance between the input points set P and the zero level set of the function f. Since the gradient can vanish or go to infinity with compactly supported basis functions, we need to use a threshold S_1 such that :

$$Err_t(f) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{f_i - f(p_i)}{\Gamma(\|\nabla f(p_i\|)}\right)^2, \tag{8.2}$$

where :

$$\Gamma(g) = \begin{cases} S_1 & \text{if } g < S_1 \\ g & \text{if } S_1 < g \end{cases}$$

8.2 Algorithm Sequence

Figure 8.2 summarizes all steps of our algorithm on a 2D shape.

1. all poles are extracted and classified from the Voronoi diagram (Fig.8.2(b));

2. poles are filtered (Fig.8.2(c));

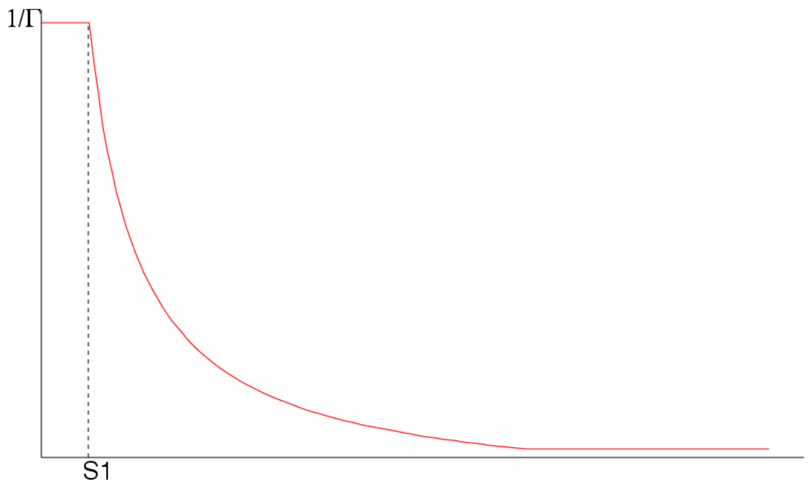


Figure 8.1: Error function. $1/\Gamma$ function.

3. poles are clustered into centers (Fig.8.2(d)) in order to selected an user defined number of centers. The set of centers is relevant if the hull of the set of inner (resp.outer) polar ball are a good approximation of the shape (Fig.8.2(e));

4. the 2D scalar function is computed and the zero-level set is extracted (Fig.8.2(f)).

As a typical example for our algorithm, we detail the timings of each reconstruction step for the *David head* model (100K points) (Fig.8.3).

1. Point insertion in the Delaunay triangulation: 6.3s;

2. Extraction of 19K poles: 3.4s ;

3. Classification (94K inside poles and 93K outside poles): 7s (Fig.8.3(b));

4. Greedy selection of 20k centers with $\rho = 0.2$: 8s (Fig.8.3(d));

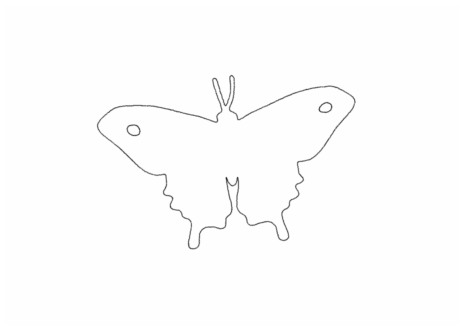
5. Assembling the linear system: 680s;
   Note that there is 98% of zero coefficients;

6. Solving the linear system: 78s.

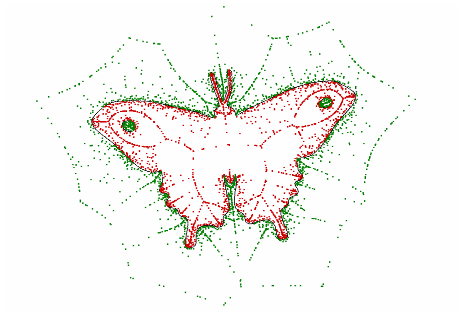
The least square mean error is about 9.4×10^{-008} and the size of the rbf file (center point - support - coefficient) is six times smaller than the mesh file (points - triangles).

In our current implementation, most of the time is spent assembling the linear system, specifically finding all pairs of centers whose supports intersect a constraint. Although the use of a 3D Delaunay triangulation avoids the naive exhaustive search (see sec.7.2).

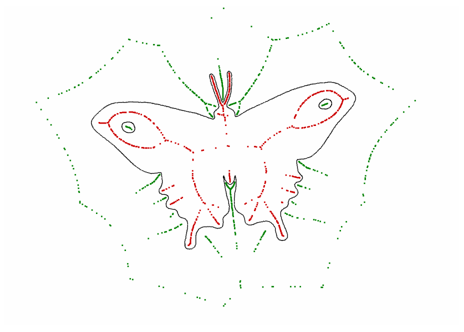
The algorithm allows to reconstruct 3D model with an hight genius like the *Filgree* model (Fig.8.4) with a genius of 65. As the RBF approach produce a watertight surface, it is well suited to fill hole, as illustrate the figure 8.5.



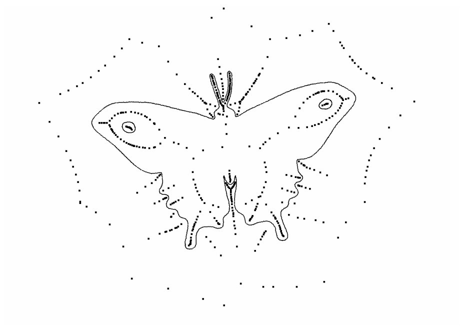
(a) 100k points measured on the surface



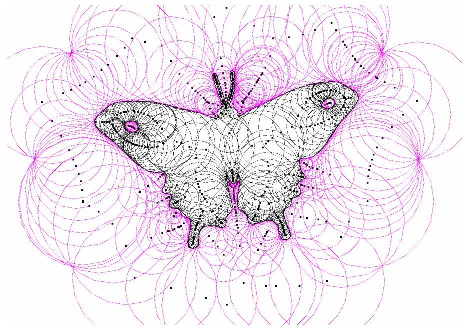
(b) 4k poles (green outside, red inside)



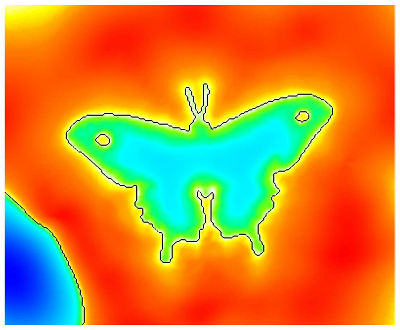
(c) 1500 filtered poles -filtering with $\lambda = 0.005$ and $\lambda_r = 0.5$



(d) 400 selected centers (red inside, green outside)



(e) 400 selected centers with the polar balls



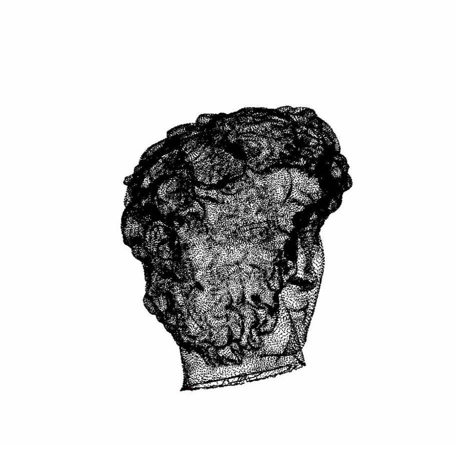
(f) Reconstructed curve (black). The color map represent the function values (cold tones for positive, hot tones for negative values and white for the zero values)

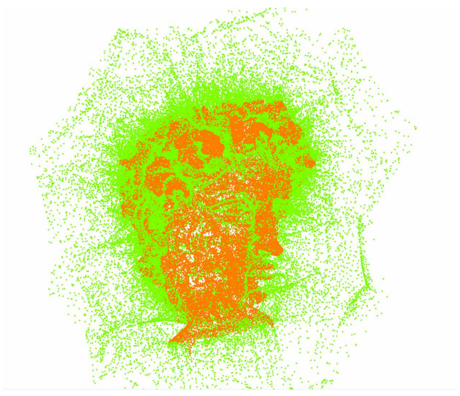
Figure 8.2: Algorithm sequence in 2D with filtering and clustering.

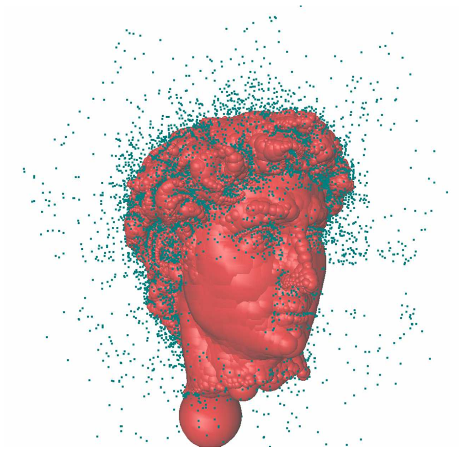
8.3 Choice of the Centers

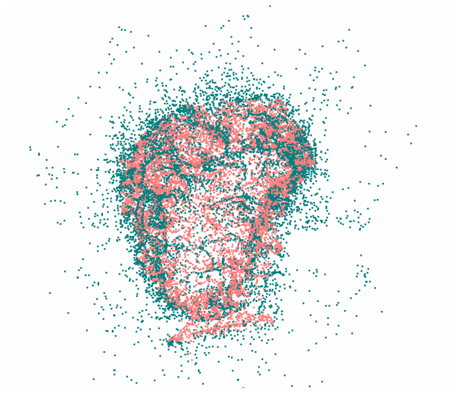
The importance of our choice for the centers is shown graphically by Figure 8.6. We plot the error against the number of centers for our method and for the common method where constraints and centers coincide. In the common method, the set of data points is subsampled and the off constraints are taken along the normals estimated at the subsampled points. Figure 8.7 illustrates several reconstructions of the Dinosaur with increasing number of centers corresponding to error plotted on the figure 8.6 (green curve).

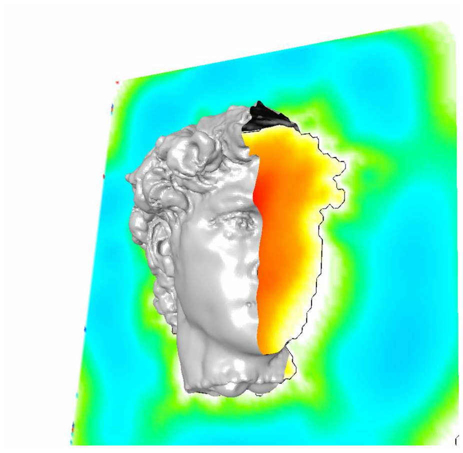
As the figure 8.8 depicts, our function is defined all over the space around the sampled shape. In contrast, when compactly supported radial basis functions are centered at the


(a) 100k points measured on the surface


(b) Poles set (green outside, orange inside)


(c) Selected centers with the inside polar balls (red inside, green outside)


(d) 20K selected centers (red inside, green outside)


(e) Reconstructed surface with a cutting plane on which the f values are represented.

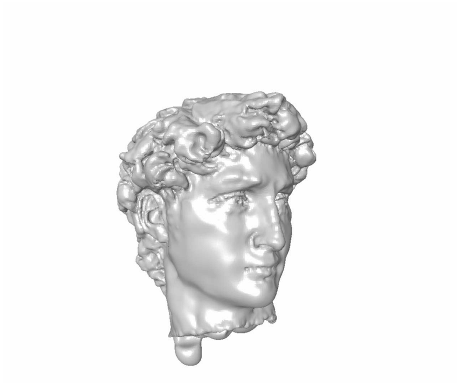

(f) Reconstructed surface

Figure 8.3: Algorithm sequence in 3D with greedy selection.

input data points, the function is only defined in a tubular neighborhood of the sampled surface.

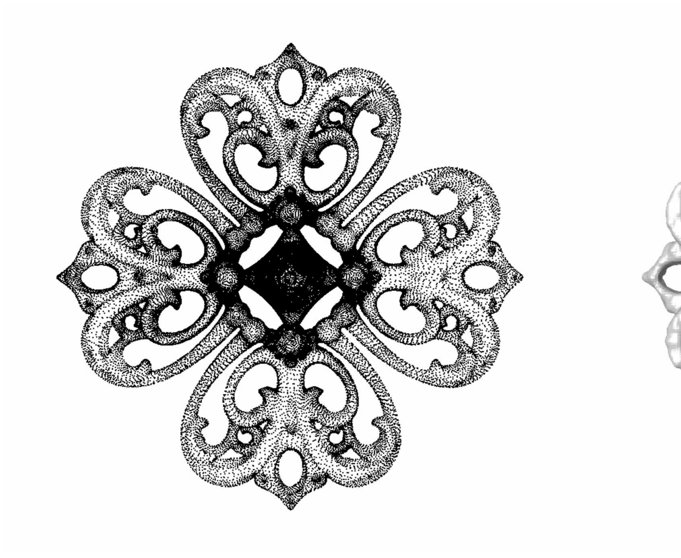
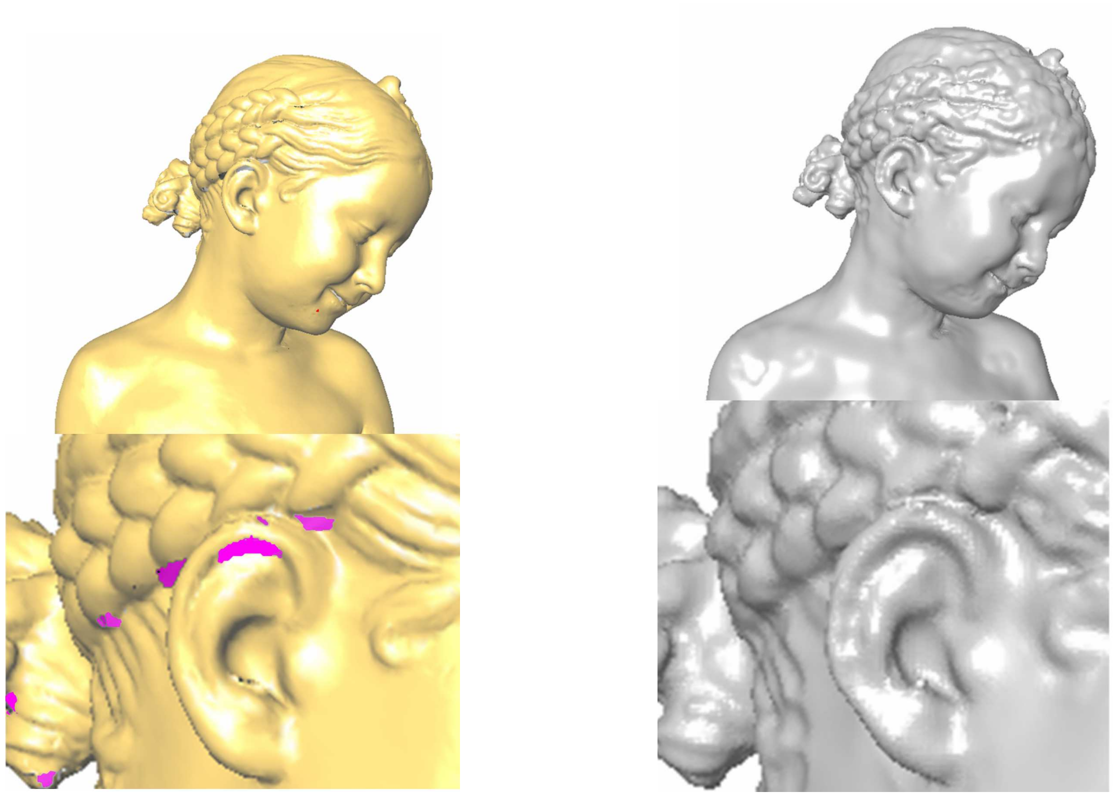


Figure 8.4: Reconstruction of the *Filgree* model (80K points) with 13K centers. Fitting accuracy: 2.8×10^{-6}. Left: the 80k input points; Right: the reconstructed surface.



(a) Original surface with holes (pink area) coming from several hidden area

(b) The reconstructed surface without any holes

Figure 8.5: Holes filling. Reconstruction of the *Bimba* model (100K points) with 11K centers obtained with clustering.

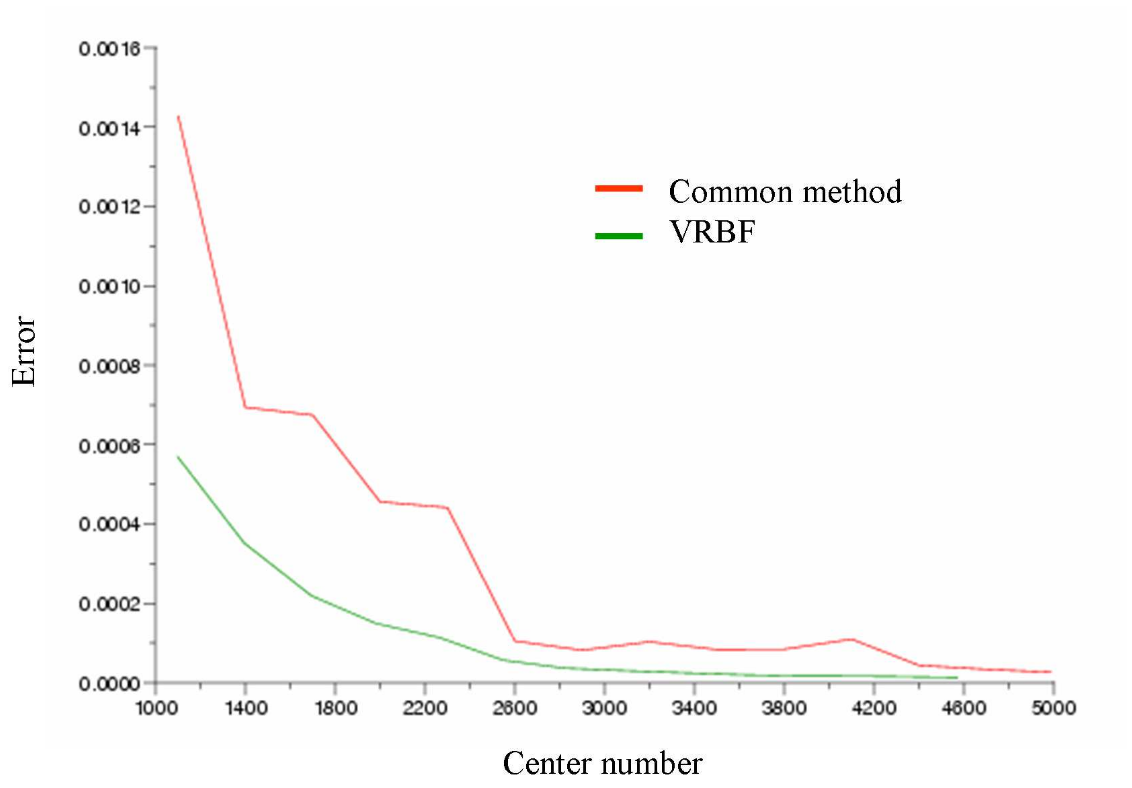


Figure 8.6: Plot of the error against the number of centers (from 1K and 5K) for the Dinosaur model (15k input points). The red curve corresponds to the common method. The green curve corresponds to our approach (with filtering and clustering).

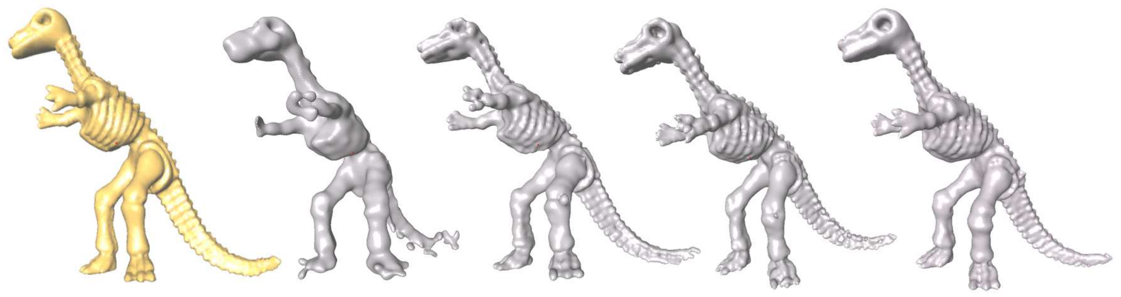


Figure 8.7: Reconstruction sequence of the Dinosaur with increasing number of centers. From left to right: original, then reconstruction with 1K, 3K, 4K and 5K centers.

8.4 Filtering and Clustering

The pole filtering step is useful to adapt the level of detail to the user-defined number of centers (Fig. 8.9), as well as to improve robustness against noise. It also shows the effect of filtering when the allocated budget of centers is low.

For the hand model, $7k$ centers do not allow to reconstruct all the details. The clustering promotes the centers associated to the small polar ball, indeed the desired centers density is proportional to the sizing field function, $sff(x)$ thus there is more centers in the area

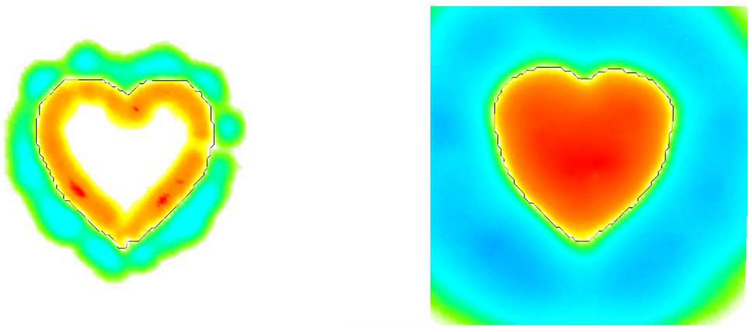


Figure 8.8: Reconstructed function. The colors represent the function values (cold tones for positive, hot tones for negative values and white for the zero values). Left: the reconstructed function for the common approach; The function does not vanish only in a tubular neighborhood of the point set. Right: the reconstructed function for our method.

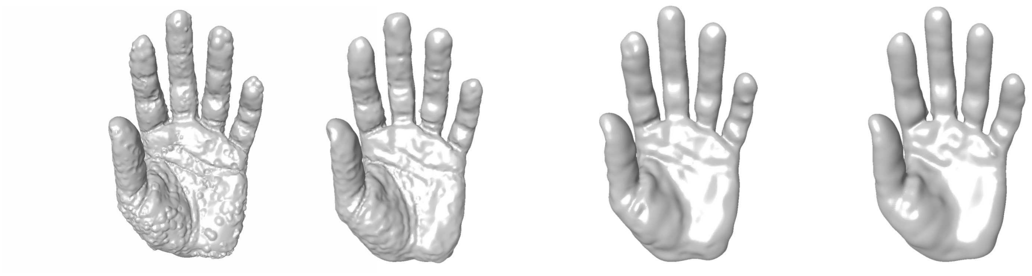


Figure 8.9: Effect of the filtering step on the Hand model (50K input points). The number of centers is $m = 7K$ obtained by clustering. From left to right : without filtering the poles filtered with $\lambda = 0.01$, $\lambda = 0.02$ and $\lambda = 0.03$ (to get a better sense of these parameters, the diagonal length of the bounding box of the input point set is 1.2 and the radius maximal value is 0.4).

where $sff(x)$ is small than in the area where $sff(x)$ is large. The figure 8.10 illustrates the center radii distribution for different filtering following by a clustering. The peak at the origin is due to the clustering which promotes the small radii. The filtering step allows to smooth down the peak and thus to obtain a distribution adapted to the center budget.

The curves plotted in figure 8.11 show the distribution of the radii for each reconstructions illustrated on the figure 8.12 and the radii distribution for all the poles without filtering and with a filtering. In the graphs 8.11(b) and 8.11(c), we can note that an uniform clustering leads to the same radii distribution than the pole radii distribution whereas a non uniform clustering produces a center set with a radii distribution independent from the poles radii distribution, i.e. from the input point distribution.
The figure 8.12 illustrate the interest of the two steps filtering and then clustering.

A uniform clustering (Fig.8.12(a) and 8.12(c)) leads to a center set with a large density in the middle of the shape where the input point density is important (Fig.8.4) . Conversely, a non uniform clustering allow to obtain a center set better distributed on the shape but if m is to small this leads to disconnected shape (Fig.8.12(b)). The filtering step

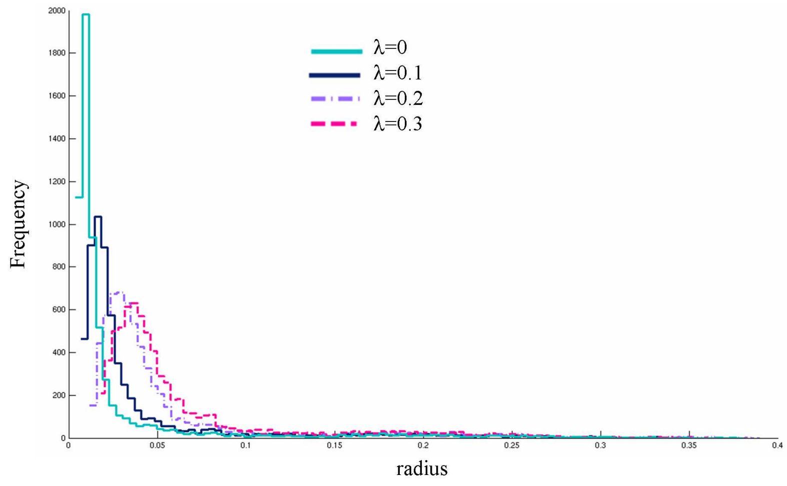
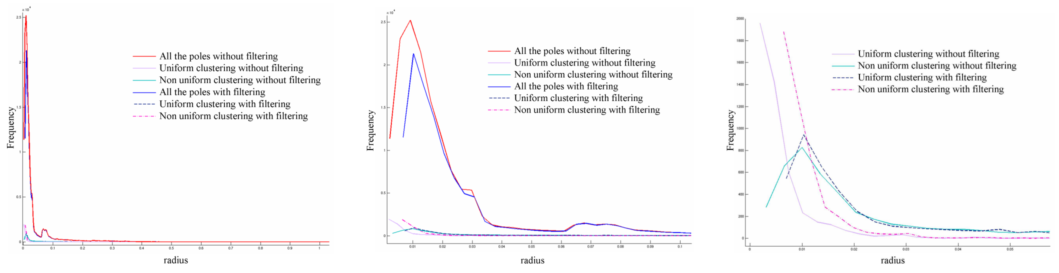


Figure 8.10: Filtering effect on the distribution of the center ball radii. The curves represent the radii distributions of 7K centers obtained by clustering with different filtering. Each curve correspond to a reconstruction on the figure 8.9



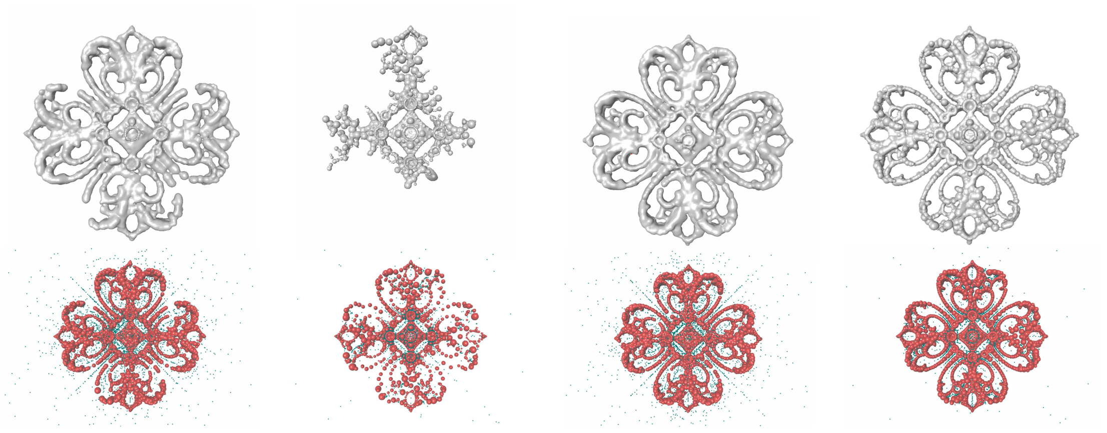
(a) Distribution of the radii (b) Detail of distribution of the radii (focus on the smallest radii) (c) Detail of distribution of the radii (focus of the 4 kind of selections)

Figure 8.11: Radii distribution for the clustering/filtering approach. The Filgree Model (50k input points and 11k poles), 5k centers are selected with clustering. The clustering step may followed a filtering or not and the clustering may be uniform or not.

handle this problem by disqualifying the centers associated with the smallest ball. The budget of centers may then used for area with larger sff values (Fig.8.12(c).

Figure 8.14 illustrates an extreme example with a substantial amount of noise due to the misregistration of three range maps. Moreover, the sampling is highly non isotropic and non uniform due to the acquisition system.

Figure 8.14 depicts the main stages of our algorithm applied to a noisy point set sampled on a hand. Although noise in the input data points leads to misclassified poles (Fig.8.14(a)), the λ-medial axis is stable under such perturbations, and theses misclassified poles are filtered (Fig.8.14(b)).



(a) Uniform clustering (b) Non uniform clustering (c) Uniform clustering with filtering $\lambda = 0.01$ (d) Non uniform clustering with filtering $\lambda = 0.01$

Figure 8.12: Filtering and clustering interest. Filgree Model (50k input points and 11k poles), 5k centers are selected with clustering. The clustering step may followed a filtering or not and the clustering may be uniform or not. Top the reconstructed surfaces. Bottom: the set of centers with the red inside polar ball.

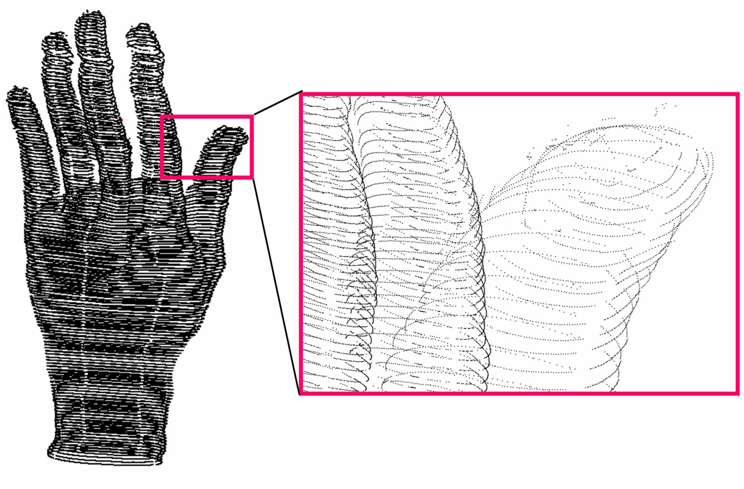
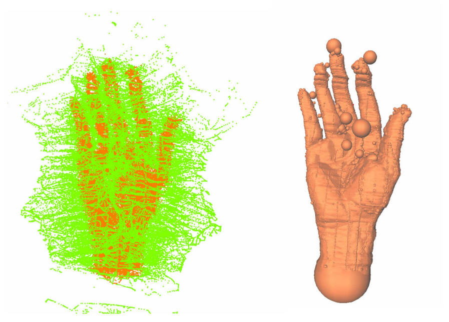


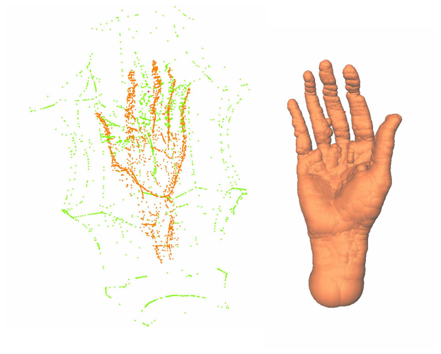
Figure 8.13: Noisy hand model (90K input points). The input points result from registering three range maps.

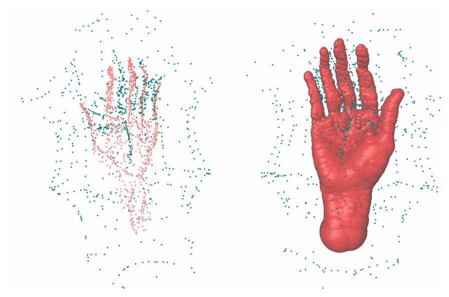
8.5 Greedy Selection

The greedy selection allows to select a set of centers well distributed. That is a centers set with a distribution independent of the sampling and adapted to the level of detail fixed by the user defined budget of centers.

The figure 8.15 illustrate the choice of the centers according to the centers budget. Figure 8.15(:left), the centers budget is sufficient and the knot is well reconstructed. Conversely, figure 8.15(:right) as the center budget is not enough, the poles corresponding to the smallest polar ball may not be selected and thus some outside poles, between the two


(a) orange inside poles with their polar balls and green outside poles (88K poles, some of them being misclassified);


(b) Filtered poles


(c) 2K centers after filtering and clustering (red inside centers with their polar balls and green outside centers

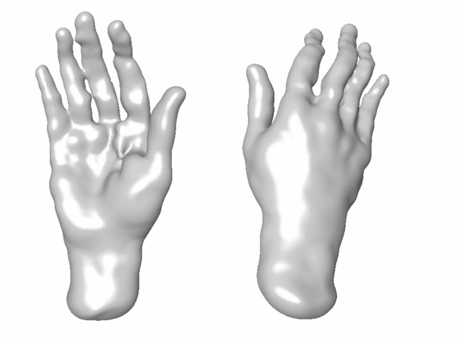

(d) reconstructed hand

Figure 8.14: Noisy hand model (90K input points).

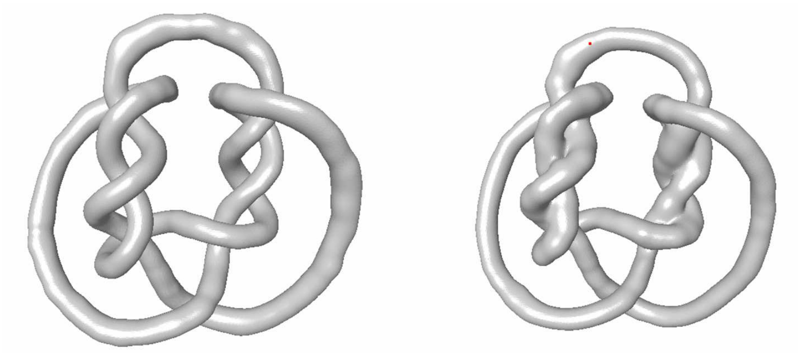
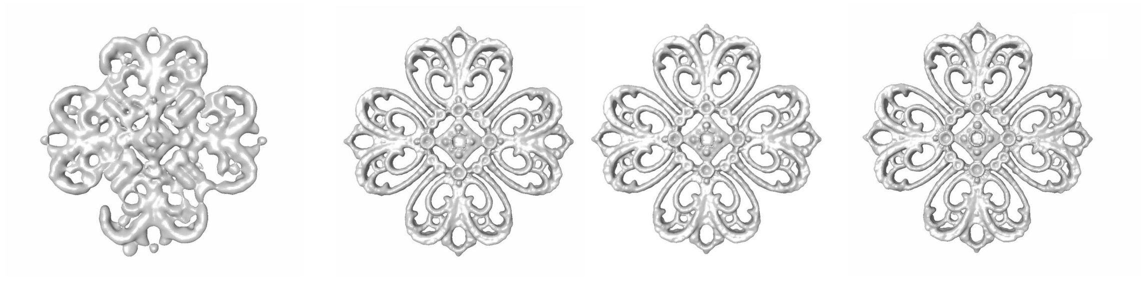


Figure 8.15: Greedy selection ($\rho = 0.6$) on the knot model (6K input points, 12K poles) Left: reconstruction with 2k centers. Right: reconstruction with 1K centers.

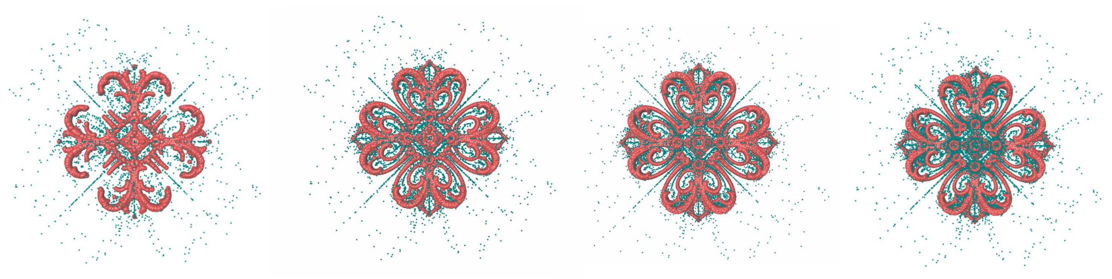
fibers, associated with small polar ball are not qualified. Thus, the fibers are not separated.

In contrast to the clustering, the greedy selection promotes the largest polar ball and the smallest polar ball are selected only if the the center budget is sufficient. The figure 8.16 show different reconstruction for a fixed overlapping rate threshold, $\rho = \frac{2}{3}$.

The figure 8.17 show different reconstruction with 11k centers with different value of



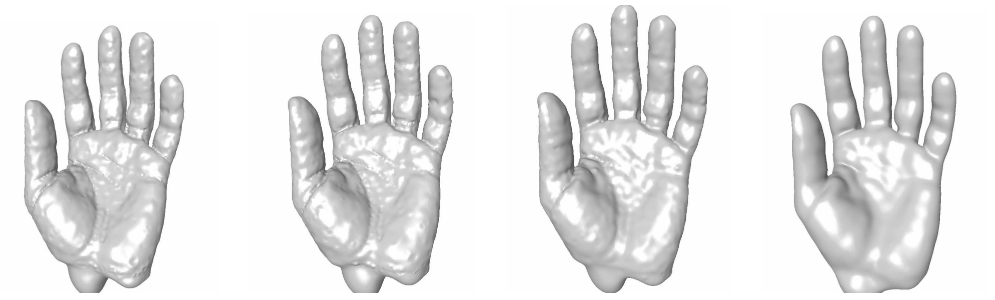
(a) Reconstructed surface



(b) Inner center polar balls and the outer centers

Figure 8.16: Effect of the greedy selection on the Filgree model (80K input points). The number of centers is form the left to the right $5K$, $10K$, 15 and $18K$. The overlapping rate threshold is $\rho = 0.6$

ρ. If the overlapping rate threshold ρ is large the reconstructed surface is smooth. Indeed,

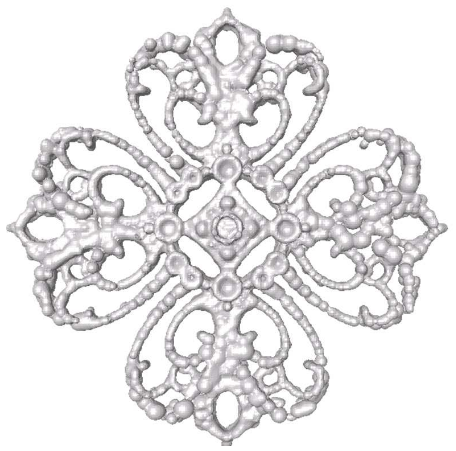


(a) $\rho = 0.1$ (b) $\rho = 0.3$ (c) $\rho = 0.6$ (d) $\rho = 0.9$

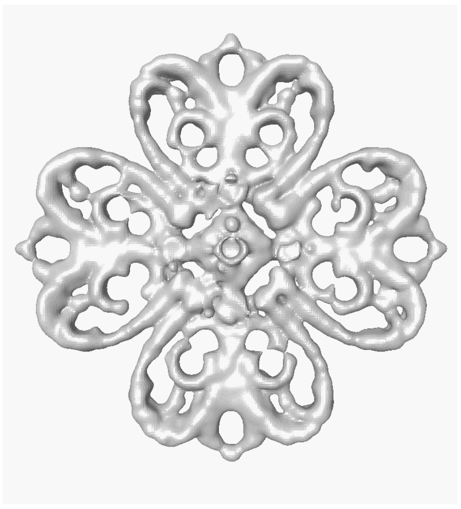
Figure 8.17: ρ effect on the greedy selection on the Hand model (50K input points). The number of centers is $11K$.

the polar ball overlapping is more important and so there is less discontinuities. However the large ρ value, the less detailed is the reconstruction. We need to find the good trade off.

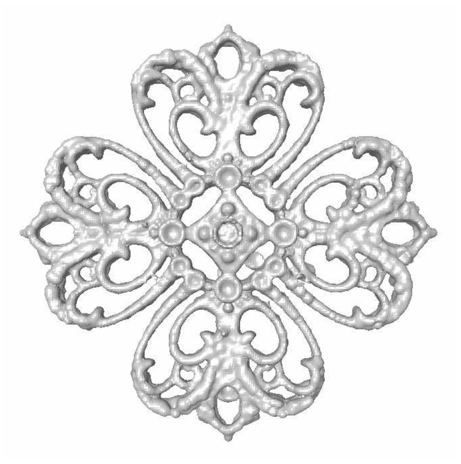
8.6 Selection Methods Comparisions



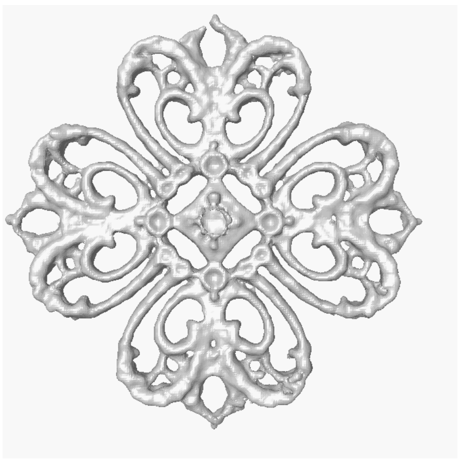
(a) Filtering and clustering $\lambda = 0.01$, $m = 5k$



(b) Greedy Selection $\rho = 0.1$, $m = 5k$



(c) Filtering and clustering $\lambda = 0.01$, $m = 10k$



(d) Greedy Selection $\rho = 0.1$, $m = 10k$

Figure 8.18:

Chapter 9

Conclusion et Perspectives

Conclusion

In this thesis, we have presented a new approach for reconstructing surfaces from scattered points, combining generalized radial basis functions and Voronoi-based surface reconstruction.
In contrast to the Voronoi-based approaches, our method creates a smooth and watertight surface, similarly to the RBF approaches. The resulting function is an approximation of the signed distance to the sampled surface defined all around the sampled shape, instead of being defined only in a small neighborhood as in previous work.
Our approach relies on a theoretically sound framework for pole extraction and λ-medial axis filtering. This framework provides us with reliable estimates of the normal at each data point, with an approximation of the distance to the sampled surface at each pole, as well as with a filtering method based on the stable λ-medial axis. As a result we can reduce the number of parameters for our algorithm to two: the number of centers, and λ, used to filter the medial axis.
In addition to filtering and clustering of the poles, we have proposed a other approach to select the basis function centers. This approach is simplest and consist in a greedy selection of the poles based on the polar balls overlapping.
The two methods produced different results as they do not promote the same kind of centers. However, for each of them we obtain a center set approximating a part of the medial axis with a density not dependent on the input points density.

Future Work

In terms of efficiency, the only stage which impairs scalability is the assembling of the final matrix. We are expect to greatly improve this aspect by an optimized implementation or using new geometric data structure.

In our study the medial axis filtering stage allows us to adapt the level of details to a user-defined budget of centers, the value for λ being fixed experimentally. In the greedy selection, the parameter ρ had to be fixed experimentally too, however the value of ρ is more intuitive than the λ value while ρ define the desired maximum overlapping rate.

In a future work, it will be interesting to investigate how to automatically adjust these parameter to accommodate for the allocated budget of centers.

Bibliography

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

[ABCO$^+$01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. *IEEE Visualization 2001*, pages 21–28, October 2001. ISBN 0-7803-7200-x.

[ACDL02] Nina Amenta, Sunghee Choi, Tamal K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry and Applications*, 12(1-2):125–141, 2002.

[ACK01] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266. ACM Press, 2001.

[AGJ02] Udo Adamy, Joachim Giesen, and Matthias John. Surface reconstruction using umbrella filters. *Comput. Geom. Theory Appl.*, 21(1):63–86, 2002.

[AH06] L. Kobbelt A. Hornung. Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In *Symposium on Geometry Processing*, pages 41–50. ACM Press, june 2006.

[AM96] D. Attali and A. Montanvert. Modeling noise for a better simplification of skeletons, 1996.

[AS00] Marco Attene and Michela Spagnuolo. Automatic surface reconstruction from point sets in space. *Computer Graphics Forum*, 19(3):457–465, 2000.

[BC00] Jean-Daniel Boissonnat and Frédéric Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 223–232, New York, NY, USA, 2000. ACM Press.

[BL88] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *ComSys*, 2:321–355, 1988.

[BL97] R. K. Beatson and W. A. Light. Fast evaluation of radial basis functions: methods for two-dimensional polyharmonic splines. 17(3):343–372, 1997.

[Bli82] James F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1(3):235–256, 1982.

[Blo94] Jules Bloomenthal. An implicit surface polygonizer. pages 324–349, 1994.

[Bra94] Jonathan W. Brandt. Convergence and continuity criteria for discrete approximations of the continuous planar skeleton. *CVGIP: Image Underst.*, 59(1):116–124, 1994.

[Buh03] M.D. Buhman. *Radial basis functions : theory and implementations*, volume 12. Cambridge monographs on applied and computational mathematics edition, 2003.

[CBC$^+$01] Jonathan C. Carr, Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 67–76. ACM Press / ACM SIGGRAPH, 2001.

[CFB97] J. Carr, W. Fright, and R. Beatson. Surface interpolation with radial basis functions for medical imaging, 1997.

[CL05] Fr&#233;d&#233;ric Chazal and Andr&#233; Lieutier. The "λ-medial axis". *Graph. Models*, 67(4):304–331, 2005.

[CSD04] David Cohen-Steiner and Frank Da. A greedy delaunay-based surface reconstruction algorithm. *Vis. Comput.*, 20(1):4–16, 2004.

[DFG99] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Review*, 41(4):637–676, 1999.

[DG03] Tamal K. Dey and Samrat Goswami. Tight cocone: a water-tight surface reconstructor. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 127–134, New York, NY, USA, 2003. ACM Press.

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

[DGH01] Tamal K. Dey, Joachim Giesen, and James Hudson. Delaunay based shape reconstruction from large data. In *PVG '01: Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*, pages 19–27, Piscataway, NJ, USA, 2001. IEEE Press.

[DTS00] H.Q. Dinh, G. Turk, and G. Slabaugh. Reconstructing surfaces using anisotropic basis functions. pages 606–613, 2000.

[Duc77] J. Duchon. Spline minimizing rotation-invariant semi-norms in sobolev spaces. In W. Schempp and K. Zeller, editors, *Constructive Theory of Functions of Several Variables*, volume 571 of *Lecture Notes in Mathematics*, pages 85–100, 1977.

[DZ03] Tamal K. Dey and Wulue Zhao. Approximating the medial axis from the voronoi diagram with a convergence guarantee. *Algorithmica*, 38(1):179–200, 2003.

[EM94] Herbert Edelsbrunner and Ernst P. M&#252;cke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 13(1):43–72, 1994.

[Far02] Gerald Farin. *Curves and surfaces for CAGD: a practical guide.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[FGK$^+$00] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr. On the Design of CGAL, a Computational Geometry Algorithms Library. *Softw. – Pract. Exp.*, 30(11):1167–1202, 2000.

[FN80] R. Franke and G. Nielson. Smooth interpolation of large sets of scattered data. *Internat. J. Numer. Methods Engrg.*, 15(11):1691–1704, 1980.

[Hay99] S. Haykin. *Neural Networks. A Comprehensive Foundation.* Prentice Hall, New Jersey, USA, 1999.

[HDD$^+$92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 71–78, New York, NY, USA, 1992. ACM Press.

[HTF01] Trevor Hastie, Robert Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations.* New York: Springer-Verlag, 2001.

[Isk04] A. Iske. *Multiresolution Methods in Scattered Data Modelling*, volume 37. Springer-Verlag, 2004.

[Kir01] Michael Kirby. *Geometric Data Analysis.* John Wiley & Sons, 2001.

[KSO04] Ravikrishna Kolluri, Jonathan R. Shewchuk, and James F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Symposium on Geometry Processing*, pages 11–21. ACM Press, July 2004.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4):321–331, January 1988.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM Press.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. volume 1, pages 281–297. University of Clafifornia Press, Berkeley, Calif., 1967.

[Mic86] C.A. Micchelli. Interpolation of scattered data: distances, matrices, and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

[Mur91] Shigeru Muraki. Volumetric shape description of range data using "blobby model". In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 227–235, New York, NY, USA, 1991. ACM Press.

[OBS03] Y. Ohtake, A. Belyaev, and H.-P. Seidel. A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions. In *SMI '03: Proceedings of the Shape Modeling International 2003*, page 292. IEEE Computer Society, 2003.

[OBS04] Y. Ohtake, A.G. Belyaev, and H-P. Seidel. 3d scattered data approximation with adaptive compactly supported radial basis functions. In *SMI*, pages 31–39, 2004.

[OBT$^+$03] Y. Ohtake, A. Belyaev, G. Turk, M. Alexa, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Trans. Graph.*, 22(3):463–470, 2003.

[OS88] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[PG89a] T. Poggio and F. Girosi. A theory of networks for approximation and learning, 1989.

[PG89b] Tomaso Poggio and Federico Girosi. A theory of networks for approximation and learning. Technical report, Cambridge, MA, USA, 1989.

[PK81] Lancaster P. and Salkauskas K. Surfaces generated by moving least squares methods. *Math. Comp.*, 37:141–158, 1981.

[RCM$^+$01] C. Rocchini, Paolo Cignoni, Claudio Montani, P. Pingi, Roberto Scopigno, R. Fontana, M. Greco, E. Pampaloni, L. Pezzati, M. Cygielman, R. Giachetti, G. Gori, M. Miccio, and R. Pecchioli. 3d scanning the minerva of arezzo. In *ICHIM (2)*, pages 266–272, 2001.

[Sch95] R. Schaback. Creating surfaces from scattered data using radial basis functions, 1995.

[Tau94] Gabriel Taubin. Distance approximations for rasterizing implicit curves. *ACM Trans. Graph.*, 13(1):3–42, 1994.

[TI04] Schilck C. Tobor I., Reuter P. Efficient reconstruction of large scattered geometric datasets using the partition of unity and radial basis functions. volume 12 of *Journal of WSCG 2004*, pages 467–474, 2004.

[TO02] Greg Turk and James F. O'brien. Modelling with implicit surfaces that interpolate. *ACM Trans. Graph.*, 21(4):855–873, 2002.

[Tol01] Sivan Toledo. *TAUCS Version 2.0*, November 2001.

[Wen95] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4:389–396, 1995.

[Wen02] H. Wendland. *Fast evaluation of radial basis functions: Methods based on partition of unity.* Number 473–483. Vanderbilt University Press, Nashville, 2002.

[Wen04] H. Wendland. *Scattered Data Approximation.* Cambridge University Press, 2004.

[Wu95] Z. Wu. Compactly supported positive definite radial functions. *Advances in Computational Mathematics*, 4:283.292, 1995.

[ZOF01] H. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method, 2001.

Voronoi, Delaunay

.1 Voronoi Diagram

Definition .1. *The **Voronoi diagram** of a point set P is a cellular decomposition of the space in regions of nearest neighborhood. Every Voronoi cell corresponds to exactly one point $p \in P$ and contains all points in the space that are closer to p than to any other points in P.*

$$V(p, P) = \{x \in \Omega : \forall q \in P \|x - p\| \leq \|x - q\|\}. \quad (1)$$

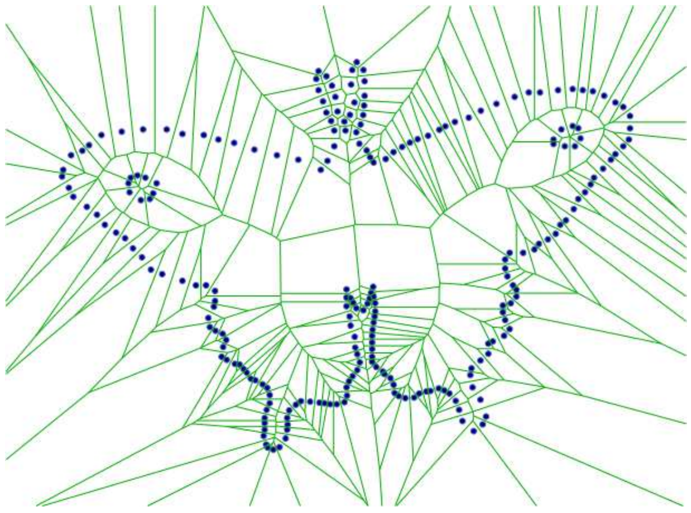


Figure 1: 2D Voronoi diagram

.2 Delaunay Triangulation

The dual of the Voronoi diagram $Vor(P)$ is called the Delaunay triangulation $Del(P)$.

Definition .2. *Delaunay triangulation:*
Whenever a collection $V_1 \ldots V_k$ of Voronoi cells, corresponding to points $p_1 \ldots p_k$, has a non-empty intersection, the simplex whose vertices are $p_1 \ldots p_k$ belongs to the Delaunay triangulation. In particular, the convex hull of four points in P defines a Delaunay tetrahedron if the common intersection of the corresponding Voronoi cells is not empty. Analogously, the convex hull of three or two points defines a Delaunay face or a Delaunay edge,

respectively, if the intersection of their corresponding Voronoi cells is not empty. Every point in P is a Delaunay vertex. The term Delaunay simplex can denote either a Delaunay vertex, edge, face or tetrahedron.

See Figure 3 for a 2D example of a Delaunay triangulation.

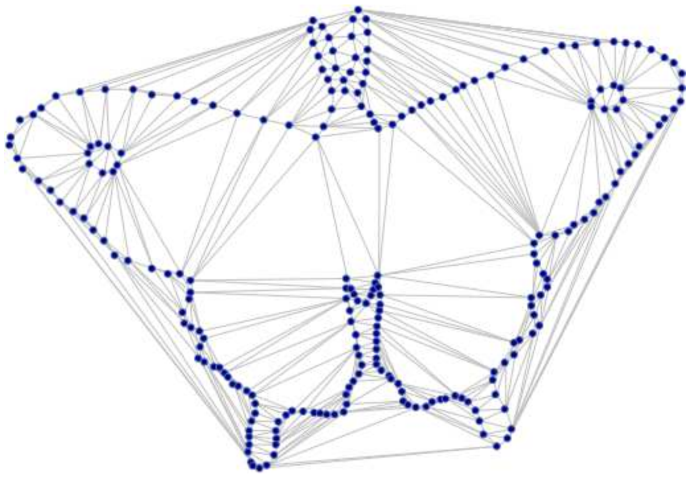


Figure 2: 2D Delaunay triangulation

.3 Restricted Delaunay

Definition .3. *The restricted Delaunay triangulation $Del_{|_S}(P)$ is the set of facets of the Delaunay triangulation whose dual edges intersect the surface.*

A ball centered on S that circumscribes a facet of $Del_{|_S}(P)$ is called a surface Delaunay ball. Its interior does not contain points of P

.4 Power Diagram and Regular Triangulation

The concepts of Voronoi Diagrams and Delaunay triangulations can be generalized to sets of weighted points. A *weighted points* $p \in \mathbb{R}^3$ is a pair of a point and a weight, (z, r). Every weighted point gives rise to a distance function, namely a the *power distance function*,

$$\pi_{(z,r)} : \mathbb{R}^3 \to \mathbb{R}, x \longmapsto \|x - z\|^2 - r \tag{2}$$

Replacing the euclidean distance by the power distance respectively yields the power diagram and the regular triangulation instead of the Voronoi diagram and the Delaunay triangulation.

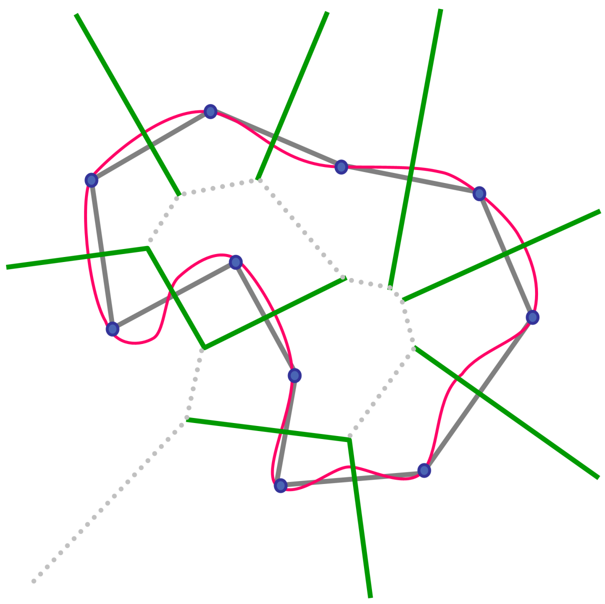


Figure 3: 2D Restricted Delaunay Triangulation.

Medial Axis and Local Feature Size

.5 Medial Axis

Definition .4. *Maximal ball:*
Let $\mathcal{O}$ be a shape $\in \mathbb{R}^3$ with a boundary $S = \partial\mathcal{O}$. A ball $\mathcal{B}$, included in $\mathbb{R}^3$, is said to be a maximal ball if there exists no other ball included in $\mathbb{R}^3$ and containing $\mathcal{B}$ (fig.4).

Definition .5. *Medial axis:*
The medial axis M of S is the topological closure of the set of points of $\mathbb{R}^3$ that have at least two nearest neighbors on S. Every point in M is the center of a maximal ball (fig.4).

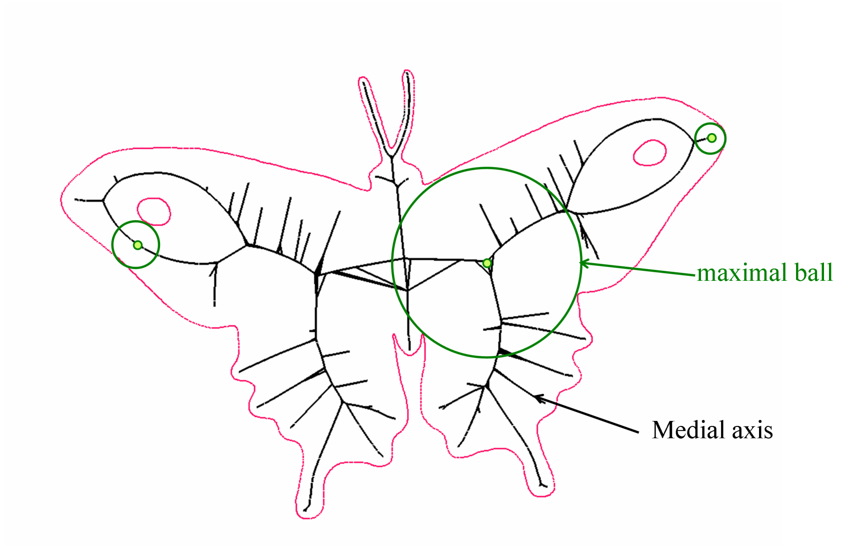


Figure 4: Inside Medial Axis. A 2D shape (red) and its inside medial axis (black).

A profusion of methods have been proposed to extract the medial axis. The exact computation of the medial axis is difficult in general. Thus the medial axis of an object has traditionally been extracted from a discrete boundary-based representation of the object. Voronoi diagrams turn out to be useful for this approximation.

Skeleton :The medial axis of a surface S is closely related to the skeleton of $\mathbb{R}^3 \backslash S$, which consists of the centers of maximal spheres included in $\mathbb{R}^3 \backslash S$. Here maximal is meant with respect to inclusion among spheres. For a smooth surface S the closure of the medial axis is actually equal to the skeleton of $\mathbb{R}^3 \backslash S$.

.6 Least Feature Size

The local feature size is a function $lfs : S \to R$ that assigns to each point in S its distance to the medial axis of S. An immediate consequence of the triangle inequality is that the local feature size of smooth surface is Lipschitz continuous with Lipschitz constant 1.

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

A sample is an **ϵ-sampling** when the distance from any surface point x to the nearest sample point is at most a small constant ϵ times the distance to the medial axis.

Abstract

This thesis considers the problem of reconstructing a surface from scattered points sampled on a physical shape. Our contribution is the development of a surface reconstruction method based on the Radial Basis Functions (RBF) approach which uses Voronoi tools in order to filter noise, reconstruct using different level of details and obtain a smaller final representation.

Recent improvements in automated shape acquisition have stimulated a profusion of surface reconstruction techniques over the past few years for computer graphics and reverse engineering applications. Data collected from scanning processes of physical objects are often provided as large point sets scattered on the surface object.

Functional based approaches where the surface is reconstructed as the zero-set of a function are standard. And the RBF approach has shown successful at reconstructing surfaces from point sets scattered on surfaces of arbitrary topology. The implicit function is defined as a linear combination of compactly supported radial basis functions.

We reduce the number of basis functions in order to obtain a more compact representation and to reduce the evaluation time. Reducing the number of basis function is equivalent to reduce the number of points (*centers*) where the functions are centered. Our aim consist in selecting a "little" set of relevant centers. In order to reduce the number of centers while maintaining decent fitting accuracy, we relax the one-to-one correspondence between the centers and the data points which is the rules in most of the RBF approaches. We depart from previous work by using as centers of basis functions a set of points located on an estimate of the medial axis. Those centers are selected among the vertices of the Voronoi diagram of the sample data points. Being a Voronoi vertex, each center is associated with a maximal empty ball. We use the radius of this ball to adapt the support of each radial basis function.

Our method can fit a user-defined budget of centers: the user can define the number of centers, i.e. the size of the representation and our algorithm will adapt the level of detail to this number using filtering and clustering or greedy selection.

Keywords

Reconstruction, Approximation, Interpolation, Regularization, Multiresolution, Implicit Surface, zero-Level sets, Radial basis functions, Voronoi, Medial axis, λ-Medial axis.

Résumé

Cette thèse s'inscrit dans la problématique de la reconstruction de surfaces à partir de nuages de points. Les récentes avancées faites dans le domaine de l'acquisition de formes 3D à l'aide de scanners donnent lieu à de nouveaux besoins en termes d'algorithmes de reconstruction. Il faut être capable de traiter de grands nuages de points bruités tout en donnant une représentation compacte de la surface reconstruite.

La surface est reconstruite comme le niveau zéro d'une fonction. Représenter une surface implicitement en utilisant des fonctions de base radiales (Radial Basis Functions) est devenu une approche standard ces dix dernières années. Une problématique intéressante est la réduction du nombre de fonctions de base pour obtenir une représentation la plus compacte possible et réduire les temps d'évaluation.

Réduire le nombre de fonctions de base revient à réduire le nombre de points (centres) sur lesquels elles sont centrées. L'objectif que l'on s'est fixé consiste à sélectionner un "petit" ensemble de centres, les plus pertinents possible. Pour réduire le nombre de centres tout en gardant un maximum d'information, nous nous sommes affranchis de la correspondance entre centres des fonctions et points de donnée, qui est imposée dans la quasi-totalité des approches RBF. Au contraire, nous avons décidé de placer les centres sur l'axe médian de l'ensemble des points de donnée et de montrer que ce choix était approprié.

Pour cela, nous avons utilisé les outils donnés par la géométrie algorithmique et approximé l'axe médian par un sous-ensemble des sommets du diagramme de Voronoi des points de donnée. Nous avons aussi proposé deux approches différentes qui échantillonnent de manière appropriée l'axe médian pour adapter le niveau de détail de la surface reconstruite au budget de centres alloué par l'utilisateur.

Mots-clés

Reconstruction, Approximation, Interpolation, Régularisation, Multirésolution, Surface implicite, Ensemble de niveaux zéro, Base de fonctions radiales, Axe médian, Voronoi, λ-Medial axis.