Voronoi Centered Radial Basis Functions

Marie Samozino

UNIVERSITÉ DE NICE - SOPHIA ANTIPOLIS

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	Rapporteur
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

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

A mes parents,
A mon frère et ma soeur

Table of 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 compact 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. The RBF approach has proved 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 to obtain a more compact representation and to reduce the evaluation cost. Reducing the number of basis function is equivalent to reducing the number of points (*centers*) where the functions are centered. Our aim consist in selecting a "little" set of relevant centers. 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. 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 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 diagram, 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 tiens à remercier en premier lieu Mariette Yvinec et Pierre Alliez, qui ont dirigé cette thèse, pour avoir su me faire bénéficier de leur expérience et de leurs compétences tout en me laissant toute liberté dans la façon de mener mes recherches.

Je remercie les rapporteurs et les membres du jury, Christophe Schick, Marc Alexa, Alexander Belyaev, Raphaëlle Chaine et Nicolas Tsingos, d'avoir accepté d'évaluer mes travaux de recherche et pour leurs remarques qui m'ont permis d'améliorer ce document.

Un grand merci à tous les membres de l'équipe Geometrica pour leur accueil, les discussions sérieuses (ou pas), les pauses cafés et leur soutien pendant ces quatres années passées à l'INRIA.

Merci a tous les amis de l'INRIA et de l'I3S (où assimilés ;-)). Pour n'oublier personne je ne citerais aucun nom mais vous vous reconnaîtrez !! Merci pour vos oreilles attentives, pour vos conseils avisés pendant les périodes difficiles.... Merci aussi pour tous les (fou) rires, les grandes discussions passées à refaire le monde, les sorties, les ragots, les matchs de foot, de basket..... Merci aussi à Fred d'avoir cru en moi et de m'avoir poussée à faire cette thèse.

Enfin, je veux remercier ma famille qui m'a soutenue et m'a permis de garder confiance en moi.

abbreviation

Voronoi Background

- M(S): Medial axis of the surface S
- MAT: Medial Axis Transform
- *lfs*: Local Feature Size

Reconstruction

- AMLS: Adaptive Moving Least Squares
- CSRBF: Compactly Supported RBF
- CAD: Computer Aided Design
- ERKPA: Enriched Reproducing Kernel Particle Approximation
- FFT: Fast Fourier Transform
- FMM: Fast Multipole Method
- GRBF: Generalized RBF
- ICP: Iterative Closest Point
- LOD: Level Of Details
- MLS: Moving Least Squares
- MPU: Multilevel Partition of Unity
- MST: Minimum Spanning Tree
- NURBS: Non Uniform Rational B-Spline
- PCA: Principal Component Analysis
- PDE: Partial Differential Equation

- RBF: Radial Basis Function
- sf: sizing function

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 information they manipulate: 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 they are used in many applications which 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. 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. For instance, tumor detection in the brain (Figure 1(top right)) requires segmentating the ill area and a geometric characterization of the corresponding 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. Another 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. Meshing rather than surface reconstruction is required 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 for art and culture diffusion in the entire world. Moreover, 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* aimed at creating a

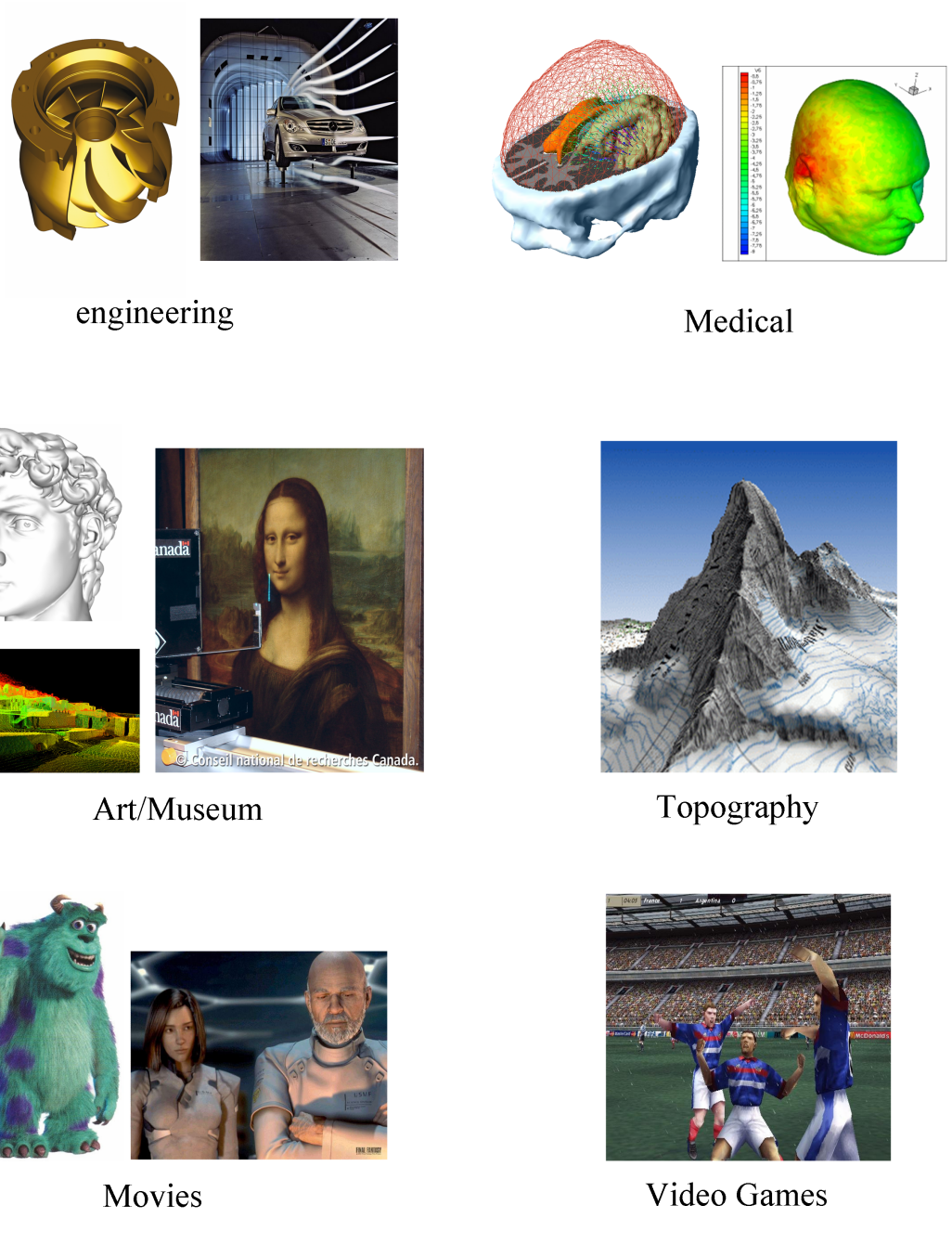


Figure 1: Application examples.

long-term digital archive of some important cultural artifacts (such as the David of Michelangelo (Figure 1(middle left));

- *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 it 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. Another application of 3D modeling is in the optimized land resource management. 3D modeling allows for working 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-world object into 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 also be 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 (Figure 2) [RCM+01]. The source emits a certain illuminant pattern and the sensor acquires the

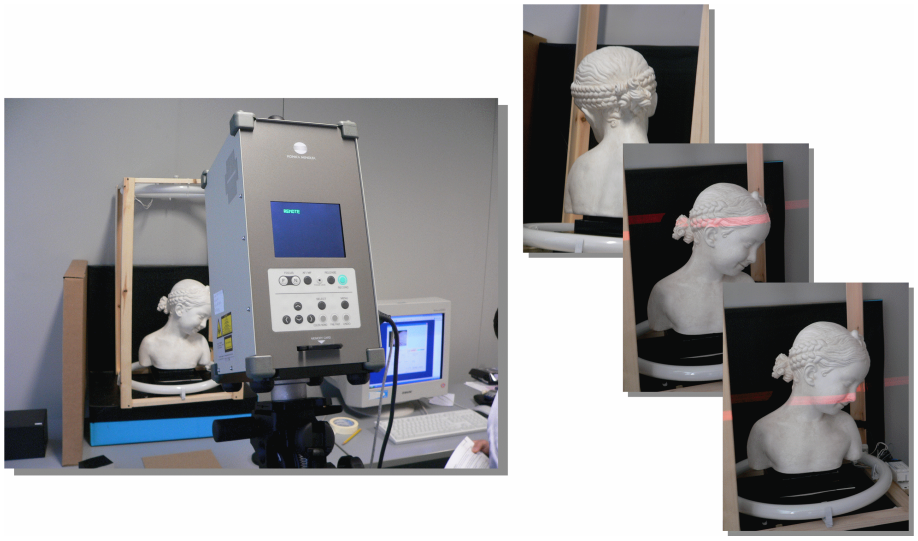


Figure 2: Bimba con nastrino. Sculpture digitized by a Minolta V910 laser scanner.

returned pattern-marks reflected by the 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 in a set of complex tasks which are commonly referred to as the *3D scanning pipeline* (Figure 3).

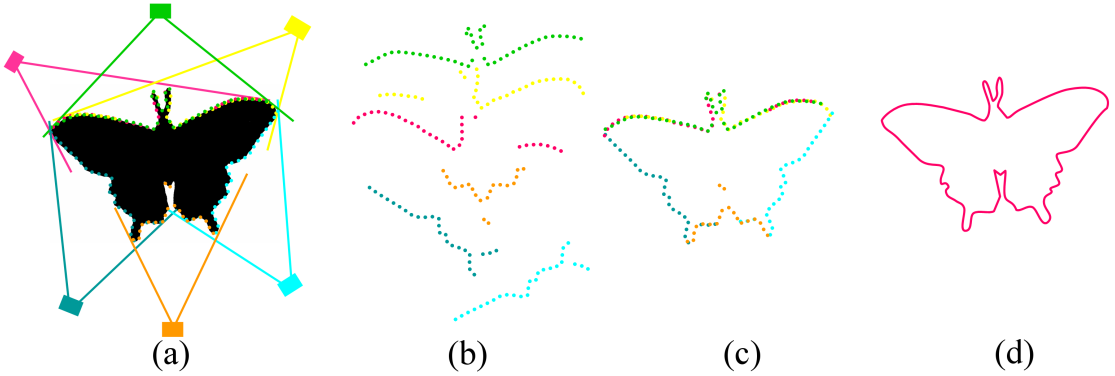


Figure 3: Scanning pipeline.
(a) Acquisition (the physical object is acquired by several scanning passes in order to cover all the surface);
(b) and (c) Registration: the range maps in their local coordinates (b) and after registration (c);
(d) Merging: the reconstructed shape.

- ≻ *Acquisition* of range images (Figure 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 maps: number, position, specifications, resolution, so as to obtain a complete sampling of the surface. The point set should cover the whole surface without 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 maps.

- ≻ *Registration* of range maps (Figure 4). Each range map is acquired independently, hence defined in a local coordinate frame which is relative to the current sensor location. The registration is the process of computing the rigid transformations (translations, rotations) to apply to each range map in order to register all the points in a single coordinate frame.
  Thus, correspondence between the different range maps 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 maps and then for all the range maps together.
  After registration, some parts of the range maps 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 maps. 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.

0.2 Goals

Our work takes place in the aforementioned *merging* step . From the point set sampled by the scanner and registered, our aim is to reconstruct the surface of the object. 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

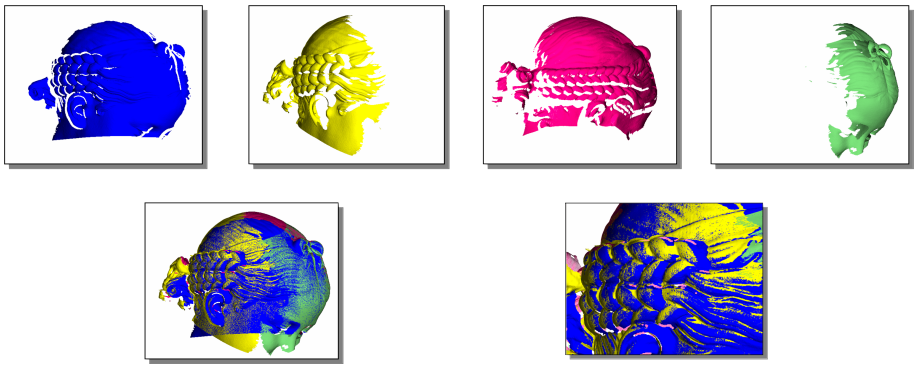


Figure 4: Registration. Top: Four range maps of the Bimba con nastrino (18 range maps were necessary to scan the sculpture). Bottom left: the range maps are merged. Bottom right: detail of the braid.

engineering applications. Data collected from scanning (Figure 2) processes of physical objects are often provided as large point sets scattered on the object surface.

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 the uncertainty in their properties (wavelength, geometry and material of lenses).

  Additional noise may arise 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 scanners operate by detecting the center of this spot, the shift causes a systematic bias in derived depth.

- ≻ *Registration*: range maps 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 maps overlap.

Moreover, selecting the n viewpoints is not easy: the overlapping rate of the range maps must be sufficient but not too redundant. For example, some surface area can not

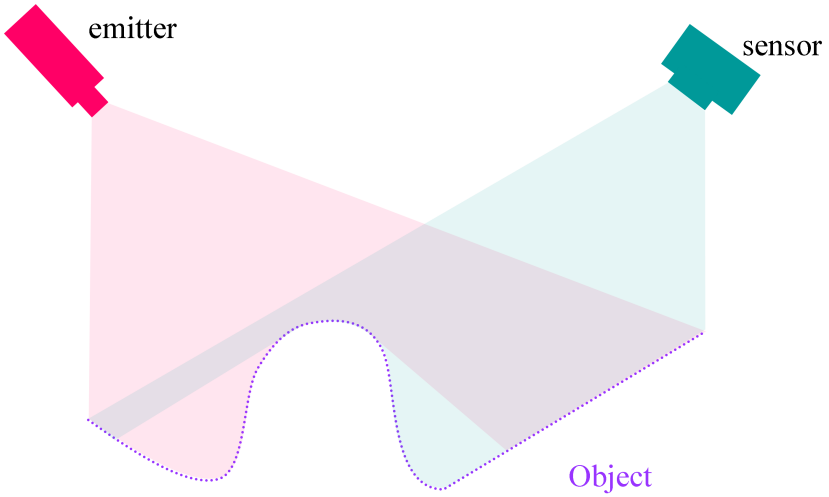


Figure 5: One of acquisition problem: hiding area. The visibility area of the emitter (pink) and the sensor/receiver (green). Some surface regions may be visible from the emitter and not visible from the receiver and vice versa.

be captured: they are visible from the emitter but not by the receiver and vice versa (Figure 5). Thus manual selection of mildly overlapping patches is non trivial. In most of the cases, a first set of range maps is measured which allow the user to decide if others scans are necessary.

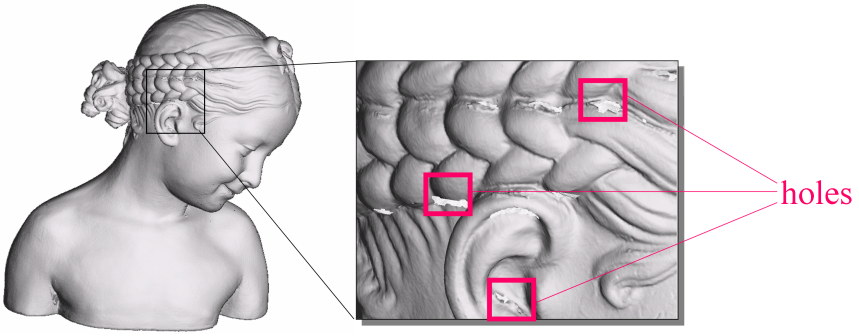


Figure 6: Reconstructing the Bimba con nastrino. The registration and fusion stages have been made with Minolta's software). Some holes appear (right) where the scan can not capture the shape (deep area)

Unadapted sampling: One important issue is to obtain a good sampling. A good sampling is a point set whose density locally adapts to the local geometry of the surface. Besides the limited accuracy of the scanner, the main difficulty comes from the fact 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 can therefore

be over-sampled (areas with a large overlapping rate between two successive range maps) whereas other parts may be under-sampled (hidden area or the surface region tangent to the laser beam) (see Figure 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 levels of detail and obtain a more compact 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 searching 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 distance to the sampled surface. This distance is defined all around the sampled shape. Furthermore, our choice for the centers allows reducing the number of centers 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 in two parts. The first part outlines the theoretical framework underlying surface reconstruction and presents a state of the art report in surface reconstruction from point sets. The second part is devoted to presenting our contributions.

Part I

State of the Art: Reconstruction from Points

Chapter 1

Introduction

Definition 1.1. *A **surface** S is a 2-manifold embedded in $\mathbb{R}^3$. Each point $p \in S$ has a neighborhood in $\mathbb{R}^3$ 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.

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 for performing various operations:

- **visualization** of the surface;
- **queries**: compute the distance to the surface or determine if a given point is in the inside or in the outside volume delimited by the surface in case of surfaces without boundary. These surfaces divide the space into two subspaces : a bounded volume tagged as inside and an unbounded volume tagged as outside.
- **modification**: a surface can be deformed, blended with another 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: visualization, geometry processing, modification or animation.

We define two main classes of surface representations: explicit and implicit. An explicit formulation describes the set of points belonging to the surface as a set of primitives. 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, triangles or elementary surface patches. 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.

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.

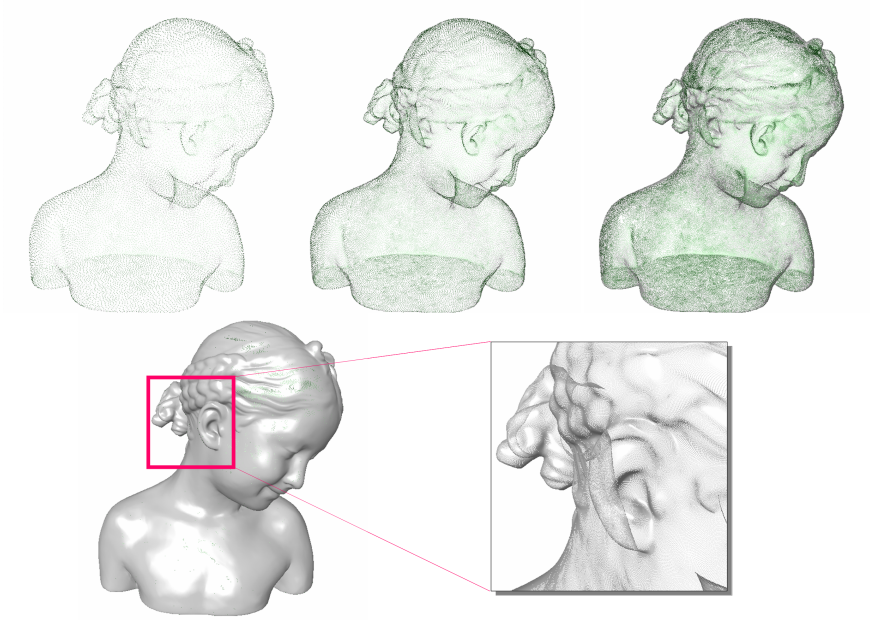


Figure 1.1: Point sample representation. Three different point sampling densities for the Bimba model. Top: three point sets: 25K, 74K and 640K points. Bottom: a point set with 2013K points and a closeup of the ear.

The point set must be dense enough to faithfully represent the shape (Figure 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 (Figure 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.

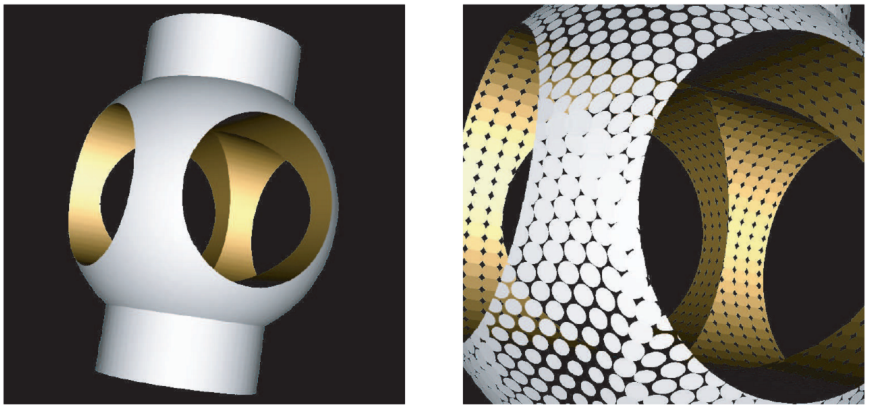


Figure 1.2: Surface splats (image taken from [KB04]).

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

Meshes

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

Triangle meshes (Figure 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 2-dimensional simplicial complex is a collection of simplices of dimension at most 2 such that faces of a simplex 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 (Figure 1.3(left)).

Parametric Surfaces

Terrain modeling is a particular case of explicit surface. The surface is represented as the graph of one function of two variables $f : \mathbb{R}^2 \rightarrow \mathbb{R}$. This definition fails to represent all the surfaces (like watertight). To overcome this problem several functions must be used to represent the surface, i.e. several surface patches are put together.

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

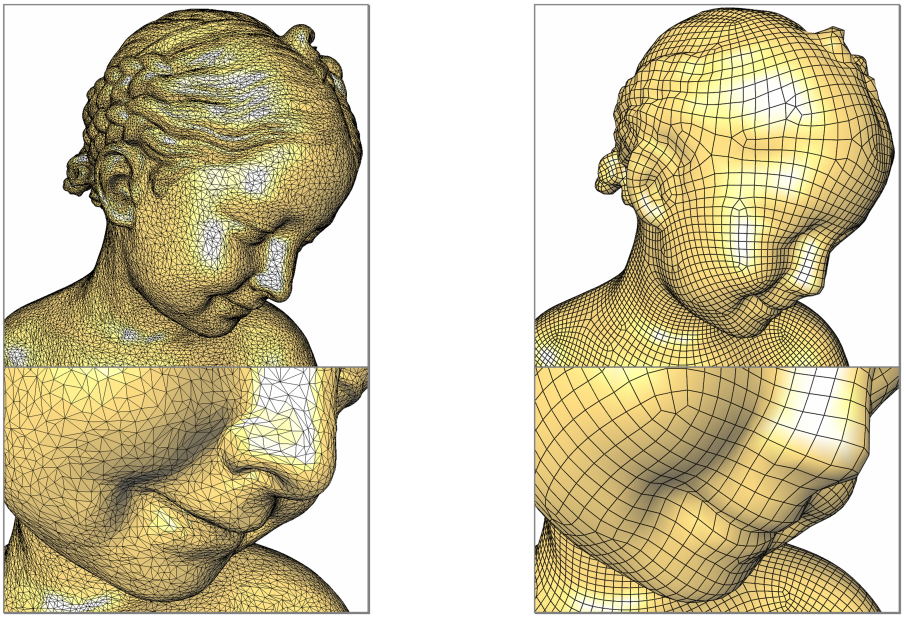


Figure 1.3: two different meshes of the Bimba. Left: Triangle mesh. Right: Quadrangle mesh.

Definition 1.3. *In a **parametric representation**, the surface is represented by a two dimensional function* $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ *which maps a two dimensional parameter domain* Ω *into* $\mathbb{R}^3$*. 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

Definition 1.4. *The surface* S' *is represented implicitly as the zero-set of a function* $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ *(Figure 1.4), i.e*

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

Note that a surface is non uniquely determined as the isocontour of a function. Indeed, several implicit functions may induce the same surface. Given a surface, a common choice for the function f is the signed distance function to the surface. The distance values are signed: positive for outside points and negative for inside points

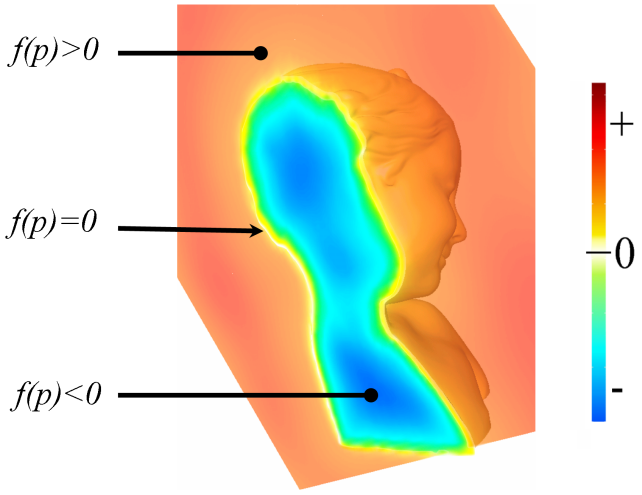


Figure 1.4: Implicit representation. The Bimba surface is defined as the zero-set of a function f, positive outside and negative inside. The colors on the cutting plane represent the function values (cold tones for negative values, warm tones for positive values and white for zero values).

1.1.3 Comparison

Location query Point location queries are easier for implicit representations than for 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 $f(p) > 0$ (Figure 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 count the number of intersections between the segment pq and 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 converting the implicit representation into an explicit one. The isovalue of the implicit function is commonly discretized into triangles using surface extraction algorithms like the *marching cubes* [LC87, Blo94] or using meshing technique such as a Delaunay-based surface mesher [BO05, RY06].

Texture mapping may be required for the visualization process. This operation is not trivial and requires 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 [CP05]. Normals can be evaluated at the vertices of a mesh, however it is an approximation [CSM03]. 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 (Figure 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' approximating the surface S.

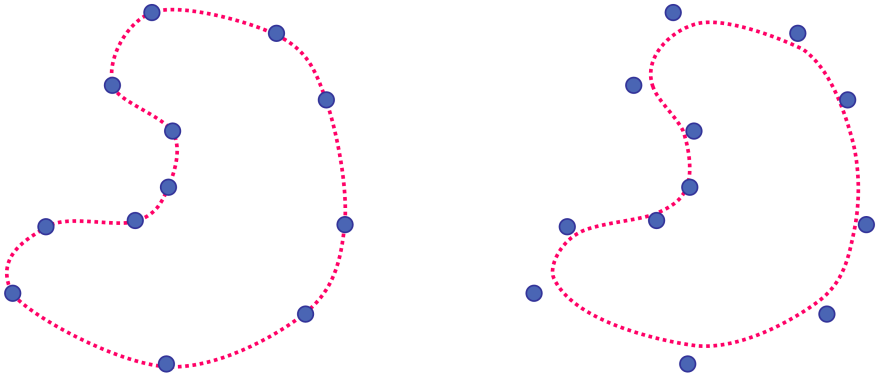


Figure 1.6: Curve reconstruction. Left : the data points are interpolated by the blue curve. Right : the data points are approximated by the blue curve.

The reconstructed surface S' may interpolate or approximate the data points P (Figure 1.6). In the interpolating case, the surface S' must pass through all data points P. Note that the solution is not unique (Figure 1.7) it depend on the chosen approach.

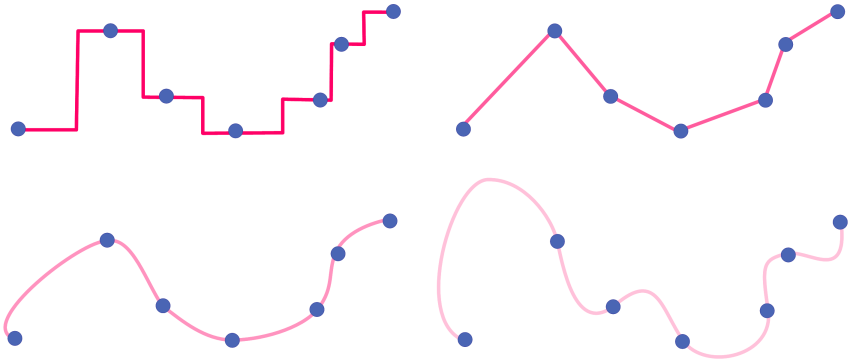


Figure 1.7: Different curves which interpolate the data points (piecewise constant, linear,.. interpolation)

Conversely, in the approximating case, the surface S' must pass close to 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 assume differential properties for the measured surface. For example, ensure that the reconstructed surface has to be smooth, 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 : Mesh based (Section 2.1) or functional based (Section 2.2).

The mesh based reconstruction algorithms establish connections between samples which are neighbors on the surface. These approaches use geometric constructions which define 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).

Several important approaches remain 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, and noisy data. The aim is to obtain a watertight surface, with a compact representation and to have an algorithm with as few parameters as possible.

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 (Chapter 1.1) used for S' depend on the chosen reconstruction method.

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 the dual *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.

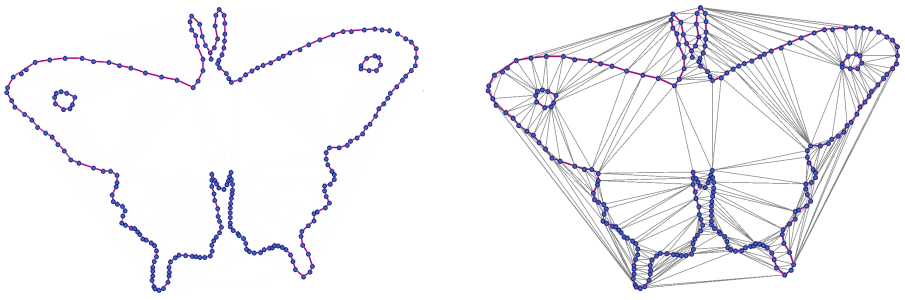


Figure 2.1: Delaunay based curve approximation. Left: the blue points are sampled on a red curve. Right:the Delaunay triangulation of the point set contains a piece-wise linear approximation of the curve

In general the Delaunay-based approaches sculpt the Delaunay triangulation of the sample points. Specifically, a subcomplex interpolating the sampled surface is extracted from the Delaunay triangulation by greedily eliminating facets from the triangulation according to geometric criteria such as the area of the triangular facets (Figure 2.1), see [CGY04] 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 (Figure 2.2). The set of real numbers α leads to a family of α-shapes.

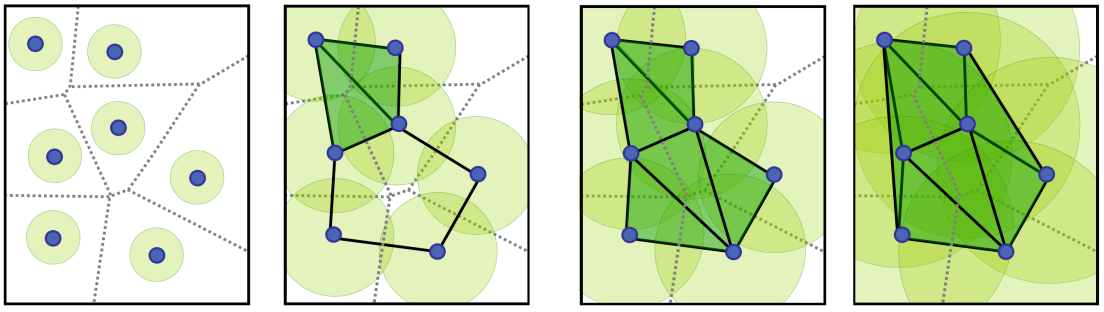


Figure 2.2: Reconstruction using α-shapes. The input points (blue) with α-balls centered at them. From left to right: the parameter α is increasing. The black edges compose the α-complex and the α-shape is the boundary of the green area.

The definition of α-shapes is based on an underlying triangulation that may be a Delaunay triangulation in case of basic α-shapes or a regular triangulation in case of weighted α-shapes.

Let us consider the basic case with a Delaunay triangulation. We first define the α-complex of the set of points S. The α-complex is a subcomplex of the Delaunay triangulation. For a given value of , the α-complex includes all the simplices in the Delaunay triangulation which have an empty circumsphere with squared radius equal or smaller than α. Here *empty* means that the open sphere do not include any points of S. The alpha shape is then simply the domain covered by the simplices of the alpha complex (see [EM94]).

In general, an α-complex is a non-connected and non-pure complex. This means in particular that the α-complex may have singular faces. For $0 \leq k \leq d-1$, a k-simplex of the α-complex is said to be singular if it is not a facet of a $(k+1)$-simplex of the complex.

The α-shapes of a set of points P form a discrete family, even though they are defined for all real numbers α. The entire family of α-shapes can be represented through the underlying triangulation of P. In this representation each k-simplex of the underlying triangulation is associated with an interval that specifies for which values of the k-simplex belongs to the α-complex. Relying on this fact, the family of α-shapes can be computed efficiently and relatively easily.

With increasing α more and more cells of the Delaunay complex appear in the α-complex (Figure 2.2). The parameter α controls the level of detail. 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 [Ede92] 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 for surface reconstruction, also called Voronoi filtering, was designed by Amenta and Bern [AB98]. This algorithm relies on the notion of the *poles* and *medial axis* (see Apendix A). 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 [Bra94]). 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.

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.

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

Definition 2.1. *The **poles** are 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 $\frac{\pi}{2}$ (Figure 2.4).*

Boissonnat and Cazals [BC00] 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 to the original surface (Figure 2.4).

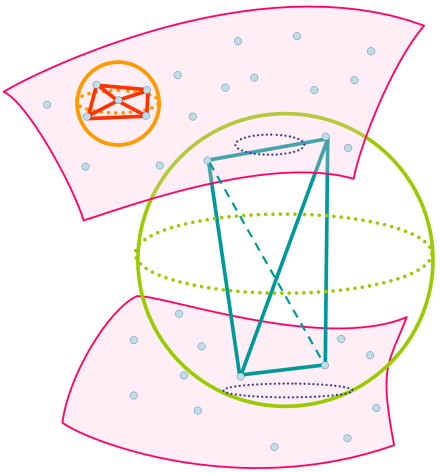


Figure 2.3: Tetrahedron configurations in 3D. The red tetrahedron corresponds to a Voronoi vertex far from the medial axis. The blue tetrahedron corresponds to a Voronoi vertex near the medial axis, i.e. a pole. The poles are guaranteed to converge to the medial axis.

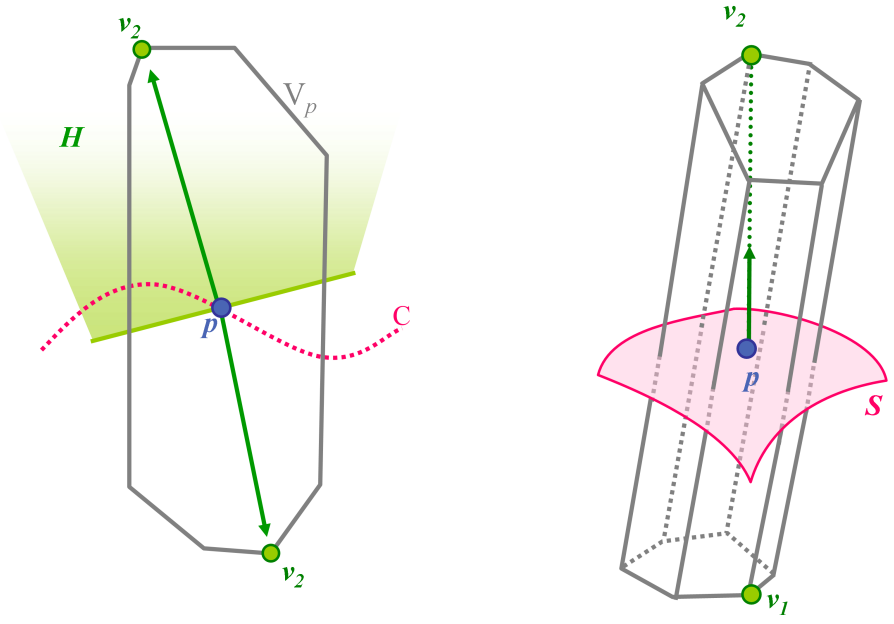


Figure 2.4: Pole. One sample point p, its Voronoi cell and its two poles v_1 and v_2. Left: 2D case. H is the opposite half space of v_1 and C the sampled curve. Right: 3D case. S is the sampled surface.

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

- ≻ Compute Voronoi diagram of P;
- ≻ For each $p \in P$ find its poles (v_1, v_2), let $V(P)$ be the set of all the poles;

- ≻ Compute the 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.

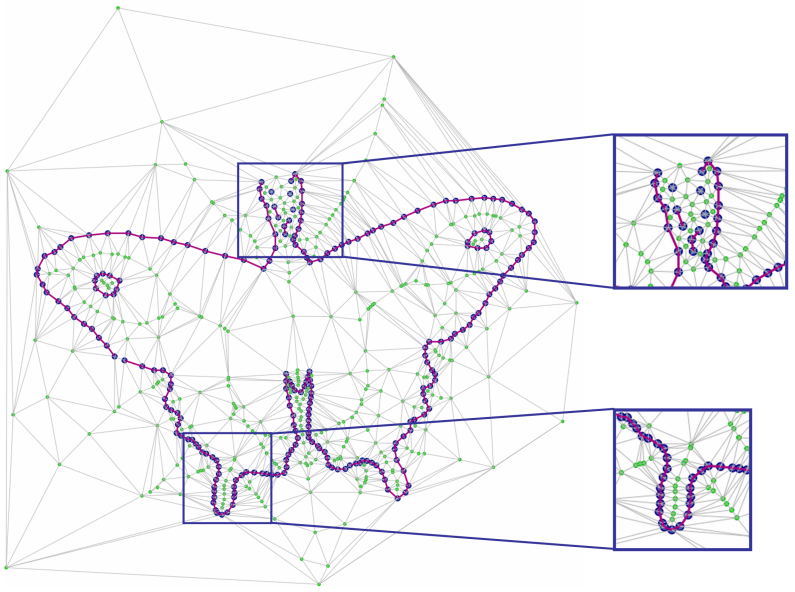


Figure 2.5: Crust in 2D. In blue the input points P, In green the Voronoi vertices $V(P)$, In gray the edges of the Delaunay triangulation of $P \cup V(P)$ and, highlighted, the Delaunay edges between points of P (red). Right: two details. Top closeup: the antennas are under sampled: the reconstruction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

In addition an approximation of the surface, the algorithm provides an approximate medial axis of S is formed by the set of Voronoi facets in T which are not in the approximate surface.

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 adapted to the *lfs*. It extends to handle noise, sharp corners, and to produce a 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 an approximation of the object as the union of the power diagram cells of the inside poles.

The algorithm provides an estimate of the interior medial axis. The power diagram define adjacencies between 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.

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 (Figure 2.7).

The Cocone algorithm computes a set of candidate triangles containing the restricted Delaunay triangulation $D_s(P)$.

Definition 2.2. *The restricted Delaunay triangulation $Del_{|S}(P)$ is the set of facets of the Delaunay triangulation whose dual edges intersect the surface (Figure 2.6).*

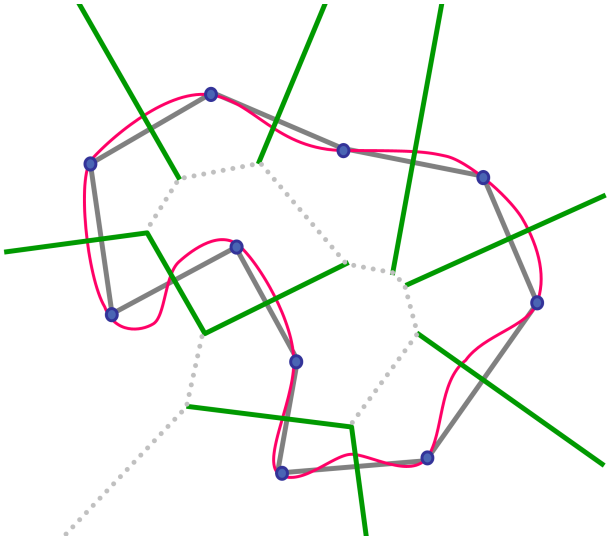


Figure 2.6: 2D Restricted Delaunay Triangulation.

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 The Cocone contains the surface restriction of the Delaunay triangulation to the union of co-cone (Figure 2.7).

Algorithm:

- ≻ Compute the Delaunay triangulation of P;

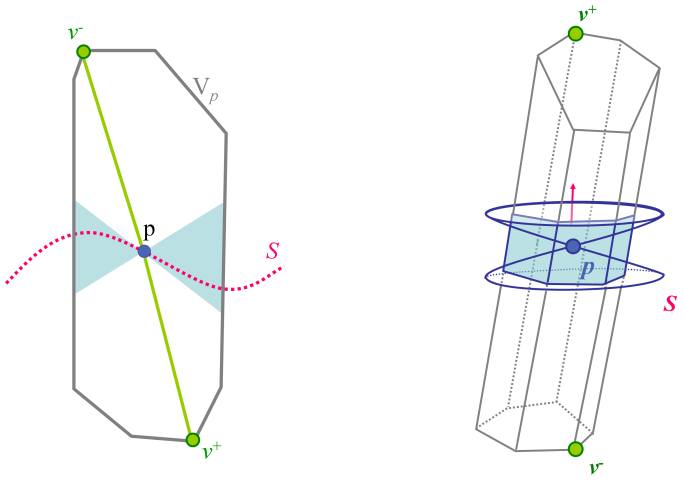


Figure 2.7: Co-cone in 2D and 3D. The co-cone of a sample point p on a curve together with the Voronoi cell of the sample point and its pole v^+.

- For every sample point $p \in P$, estimate the normal to S at p using the pole of the Voronoi cell V_p in $V(P)$ (like in Section 2.1.2);
- For each sample point p select a set of candidate triangles. If the sampling density is sufficiently high, the 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 and there may be holes.

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 then 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.

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 (Figure 2.8). 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 inside maxima.

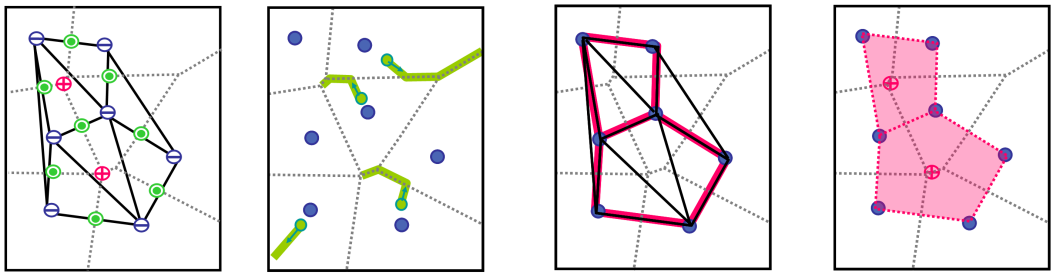


Figure 2.8: Flow complex. From left to right: 1. The local minima $\ominus$, saddle points $\odot$ and local maxima $\oplus$ of the distance function induced by the sample points (local minima). 2. Some orbits of the flow induced by the sample points (blue). 3. The stable manifolds of the saddle points. 4. The stable manifolds of the local maxima.

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 (Figure 2.9) 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.

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 2.9). 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 subgraphs containing respectively the inside and the outside poles.

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

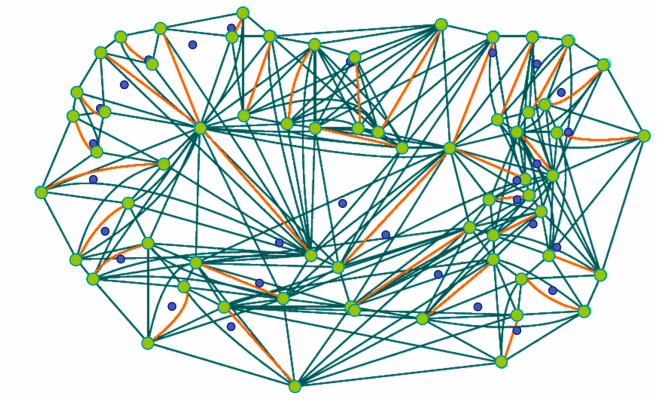


Figure 2.9: Pole graph. The negative weighted egdes (orange) and the positives ones (dark green). The weight is large if the maximal balls intersect deeply. (The input point (blue) and the poles (green))

2.2 Functional Based Surface Reconstruction

The surface S may be defined implicitly by a function (see Section 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 a signed distance from x to the object surface S.

S is unknown, the distance function cannot be computed exactly, thus an approximation must be performed.

Usually the signed distance function is approximated as a linear combination (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 the zero level set. The distance function may be defined over the entire space or in a neighborhood of the point set.

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 the distance function (Section 2.2.6).

Functional based approaches are robust when the data points are unorganized and non uniform. However their computational requirements 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] have 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) of the k nearest neighbors of p_i (Figure 2.10). 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 should ideally be proportional to the curvature in some

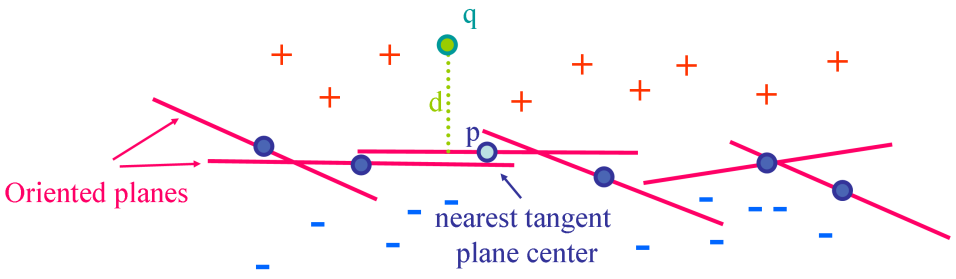


Figure 2.10: Signed Distance function. For a given point q, its nearest neighbor p in the set of input point (blue) is found then the distance d to the estimated tangent plane in p is computed.

sense, Curless & Levoy [CL96] have proposed a variant such that the distance function is defined as a weighted sum of distance functions defined for each range map:

$$D(x) = \sum_{i=1...k} \frac{w_i(x)d_i(x)}{\sum_{i=1...k} w_i(x)} \tag{2.3}$$

where, $d_i(x)$ are the signed distance and $w_i(x)$ are the weight functions from the i^{th} range maps.

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 [BC00], Boissonnat an Cazals proposed a smooth surface reconstruction via *natural neighbor interpolation* of the signed distance functions to tangent planes. 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 point $x \in \mathbb{R}^3$, the distance function is computed as:

$$f(x) = \sum_{p_i \in P} \mu_i(x)(x - p_i).\vec{n} \tag{2.4}$$

where $\vec{n}$ is the normal to the surface at p_i and $\mu_i(x)$ is the natural coordinate of x.

Given a Delaunay triangulation, the neighborhood of a vertex is naturally defined as the set of vertices connected to that vertex. This information is of combinatorial nature and can be made quantitative using the so-called *natural coordinates*. The natural coordinate of a point x according to a point $p - i$ of P is the normalized measure of the region of withdrawn from p_i if x is added to the Voronoi diagram (Figure 2.11)

2.2.2 Surface Fitting

Another approach is to perform some type of fitting to the data using a polynomial [LBC96] or an algebraic surface [GO93] to the data.

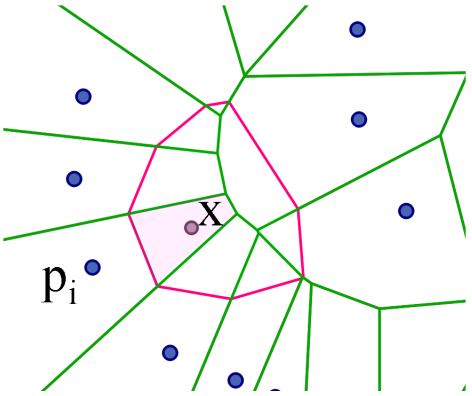


Figure 2.11: Natural Neighbors. Point x has 8 natural neighbors. The red cell is created when the point x is added to the Voronoi diagram (green). The normalized surface of the pink area correspond to the coordinate of x according to p_i

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. Specifically, the function/model f fit 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:

$$f^* = \arg\min_f (\sum_{i=1}^{n} (f_i - f(p_i))^2). \tag{2.5}$$

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

$$E(f) = \sum_{i=1}^{n} (f_i - f(p_i))^2 + \lambda \|D(f)\| \tag{2.6}$$

where λ is the regularization parameter which allows tuning 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 uniform, for example the required smoothness is the same for over all the surface.

In the *Multilevel Partition of Unity implicit* algorithm (MPU) [OBT+03] a quadric is fitted preserving sharp features by using a multilevel partition of unity and three kinds of fitting model. The surface is defined locally with quadric functions which 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:

$$f(x) = \sum_{i=1}^{M} w_i(x) f_i(x). \tag{2.7}$$

The weights w_i are smooth functions and sum up to one everywhere on the domain. They 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:

$$w_i(x) = \frac{W_i(x)}{\sum_{j=1}^{n} W_j(x)}. \tag{2.8}$$

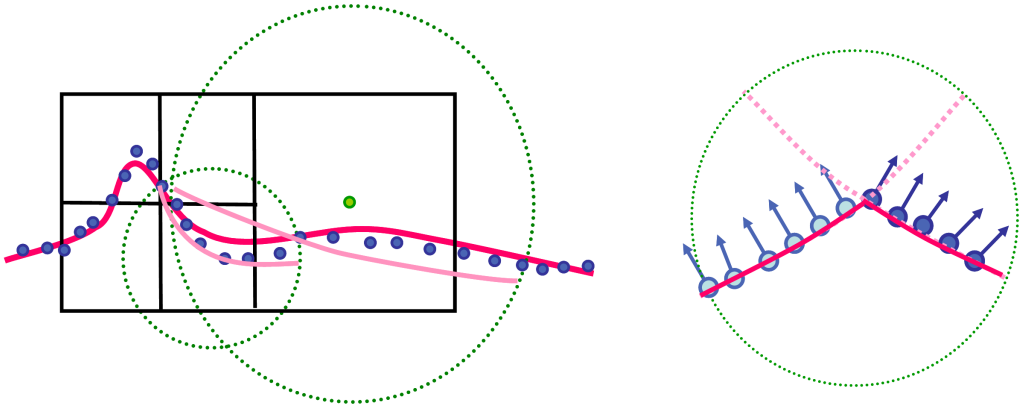


Figure 2.12: Multilevel partition of unity implicit. Left: an octree is computed according to the distribution of the data points (blue). For each cell of the octree, a piecewise quadric surface (light pink) is fitted to the points contained in a ball centered at the cell. The type of piecewise quadric fitted is chosen according to the distribution of the normal and 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.

In MPU approach, the partition on subdomains is induced by an octree. The fitting strategy is adapted to each cell of the octree. According to the number of points and the normal distribution, 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 normal distribution 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 normals is performed (Figure 2.12).

2.2.3 Moving Least Squares

The idea of the Moving Least Squares (MLS) approach [PK81] comes from the least squares technique to fit a surface to a set of points described in Section 2.2.2.

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

Definition 2.3. *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 $\psi : \mathbb{R}^3 \to \mathbb{R}^3$:*

$$S = \{x \in \mathbf{B} : \psi(x) = x\}, \tag{2.9}$$

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

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

$$\mathbf{B} = \left\{x \in \mathbb{R}^3 : \|x - p_i\| < r_{\mathbf{B}}\right\}. \tag{2.10}$$

Let x be a point in a neighborhood of P. A local reference plane H_x for x is determined

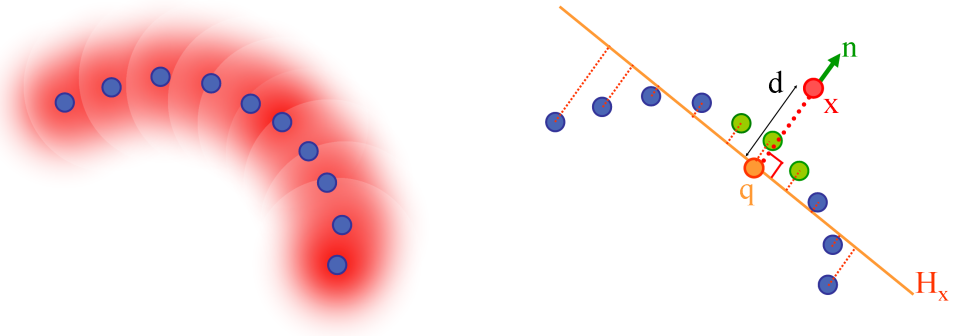


Figure 2.13: Moving least squares. Left: the tubular neighborhood $\mathbf{B}$ of the blue point set. Right: A local reference plane H_x for the points x (red) is determined by minimizing a local weighted sum of squared distances of the points p_i (blue and green) to the plane H_x. q (orange) is the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

by minimizing a local weighted sum of squared distance of the points p_i to the plane H_x:

$$E_{\vec{n},q}(x) = \sum_{i=1}^{N} (\vec{n} \cdot (p_i - q))^2 \, \theta(|p_i - q|), \tag{2.11}$$

where $\vec{n}$ is the unit normal to H_x, $q = x + d\vec{n}$ is the projected point of x on H_x, d is the distance from x to the plane and θ is a radially symmetric, positive and monotically

decreasing weight function. The algorithm iteratively minimizes the energy function $E_{\vec{n},q}$ to find $\vec{n}$ and q in order to determine the projector Ψ such as $\Psi(x) = q$.

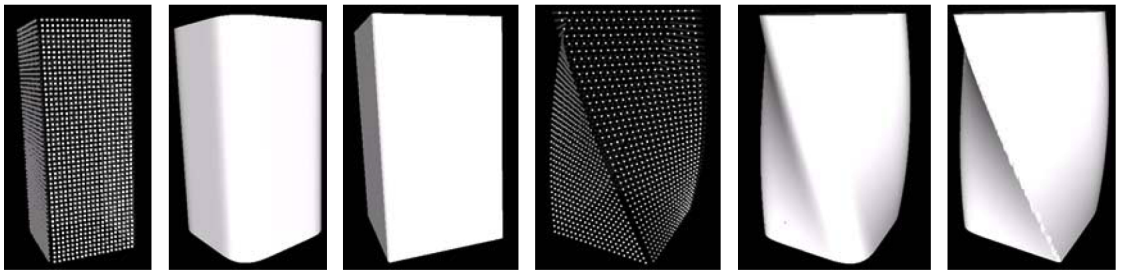
Several definitions have been proposed to define the weight function θ. The projection algorithm is also derived in several variants.

Alexa et al. [ABCO+01] proposed to add a polynomial fitting step. 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 (Figure 2.13(right)). The weighted energy is defined by:

$$E = \sum_{i=1}^{N} (f(x_i, y_i) - d_i)^2 \; \theta(|p_i - q|), \tag{2.12}$$

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.

In the Adaptive MLS (AMLS) [DS05], 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.



(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.

In [RJT+05], Reuter et al. have 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. The 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-specified support size allows for controlling the influence domain of the sharp feature (Figure 2.14).

2.2.4 Radial Basis Functions

About 20 years ago, in an extensive survey, Franke [FN80] identified Radial Basis Functions (RBF) as a 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 basis function

$\Phi : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}$, as a linear combination

$$f(x) = \sum_{j=1}^{m} \alpha_j \Phi(x, c_j), \tag{2.13}$$

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 invariance according to rotation and 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 algorithms have been proposed [CFB97, DTS01, OBS03, TI04, CBC+01, Sch95, BL97, Buh03, MYC+01, OBS03, OBS04, Wen02, Wen95, Wu95]. As our approach is based on RBF, we will review the corresponding literature in Chapter 3.

Our RBF approach locates the center points c_i on some Voronoi vertices called poles. As these poles lie close to the medial axis, our approach is related to the skeleton based methods, described as follows.

2.2.5 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 primitive is associated to a potential function, $v(x)$, which decreases according to the distance from x to the primitive.

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 functions D_i centered at each point p_i of the skeleton (2.14).

$$D_i(x) = b_i * exp(-a_i \cdot r_i(x)^2), \tag{2.14}$$

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

$$f(x) = \sum_{i=1}^{N} D_i(x). \tag{2.15}$$

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 (Figure 2.15):

$$E = \sum_{i=1}^{n} (f(p_i) - iso)^2, \tag{2.16}$$

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

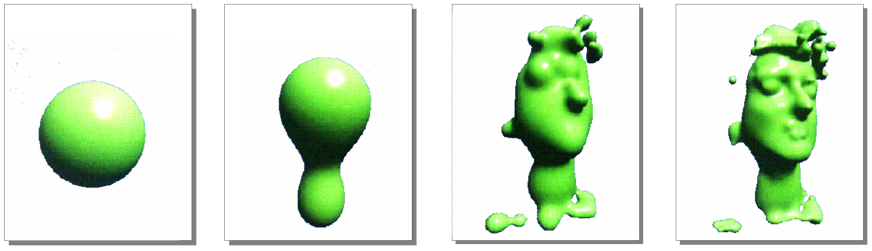


Figure 2.15: Blobby Model. The transformation of a "Blobby Model" with the number of primitives N. From left to right : $N = 1, 2, 35, 243$

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 [TBC95] 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.

2.2.6 Indicator Function

In [KBH06], Kazhdan has developed a surface reconstruction technique which fits an indicator function to a point set with oriented normals (Figure 2.16). Let $\mathcal{O}$ be the object such that S is its boundary, the indicator function is defined as:

$$\chi_{\mathcal{O}}(p) = \begin{cases} 0 & \text{if p} \notin \mathcal{O} \\ 1 & \text{if p} \in \mathcal{O} \end{cases} \tag{2.17}$$

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 from the input normals. The

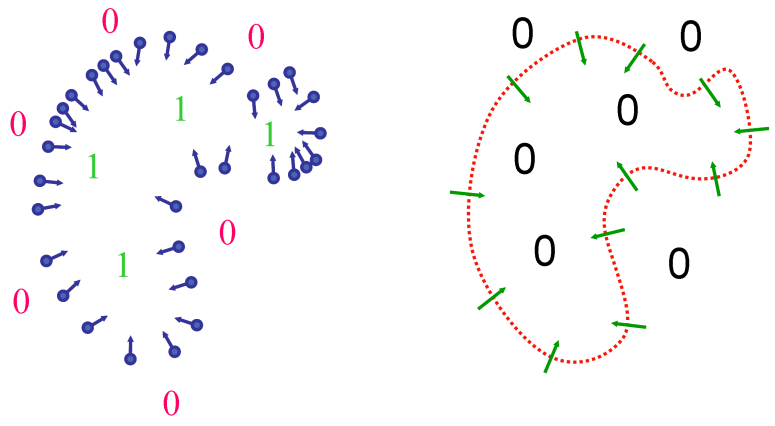


Figure 2.16: Poisson surface reconstruction. Left: the input point with theirs oriented normals. The value of the indicator function $\chi_{\mathcal{O}}$ are 0 outside and 1 inside the object. Right: the indicator gradient $\nabla\chi_{\mathcal{O}}$ is non zero in the neighborhood of the input points.

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:

$$\nabla f(x) = \vec{n}(x) \quad \forall x \in \mathbb{R}^3. \tag{2.18}$$

By using the divergence operator, the equation (2.18) becomes:

$$\Delta f(x) = \vec{\nabla} \bullet \vec{n}(x) \quad \forall x \in \mathbb{R}^3. \tag{2.19}$$

The divergence of the vector field is computed on the cells of the octree and the equation (2.19) is solved by a *Fast Fourier Transform* (FFT) and a multiscale linear solver.

In [SBS06], 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 performs 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. The octree cells are subdivided 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 close 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.

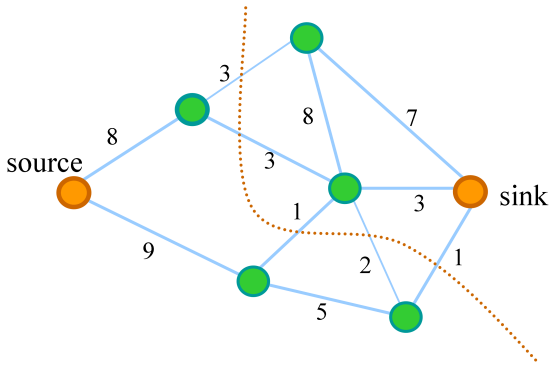


Figure 2.17: Min cut in 2D. Given a graph with valued edges (e, w_e), find min cut between source and sink.

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 2.17. 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:

$$cost = \sum_{e \in C} w_e. \tag{2.20}$$

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 (Figure 2.18) in which normal orientation is not required. Their approach uses an unsigned distance function, hence does require any local surface orientation.

Figure 2.18 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 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

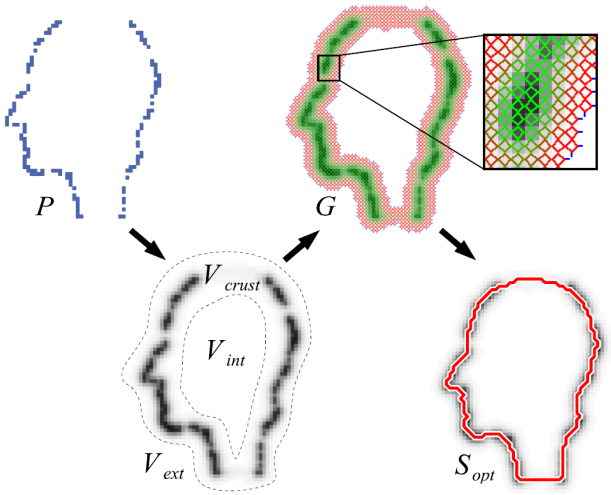


Figure 2.18: Reconstruction process in 2D

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 voxel with high confidence values are associated to small edge weights and vice versa. The boundaries are connected to 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

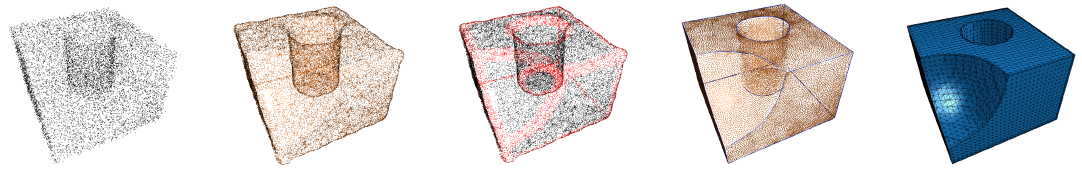
Statistical approaches were proposed to deal with noisy data and to reconstruct a piecewise smooth surface.

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

The Bayesian statistics approach to this problem is to define a probability distribution $P(S)$ over the set of all possible original scenes. The Bayes rule can then be applied 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:

$$P(S|D) = \frac{P(D|S)P(S)}{P(D)} \tag{2.21}$$

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 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



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

Figure 2.19: Bayesian point cloud reconstruction.

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 (2.21) is only a normalization constant, not depending on S, it is sufficient to compute:

$$S_{MAP} = \arg\max_S P(S|D) = \arg\max_S P(D|S)P(S) \tag{2.22}$$

The challenge is therefore to define a probabilistic model both for the measurement process and for the prior. The *Bayesian Point Cloud Reconstruction* algorithm [JWB+06] reconstructs both topology and geometry in form of a well-sampled point cloud with noise removed.

Let D be the data points sampled on a surface S. We assume that the data are entangled with noise, and that points have been lost in the acquisition process. The algorithm tries to find the reconstructed surface $\tilde{S}$ that maximizes the posterior probability $P(\tilde{S}|D)$ by numerical optimization.

The algorithm proceeds as follows: An estimate of the reconstructed point cloud $\tilde{S}$ is initialized with the original measurement points D and additional points distributed randomly in the neighborhood of the points from D, with a probability inversely proportional to the local sampling density of D. This first step is called *smoothing* (Figure 2.19(b)) while a polynomial is fitted to the data points D to remove noise and to minimize the curvature.

The algorithm then tries maximizing the a posteriori probability neglecting the discrete components of the model. This goal is achieved by numerical descent: the gradient of the posterior probability $P(\tilde{S}|D)$ is computed for the candidate point cloud $\tilde{S}$ with respect to the positions of all points, and a gradient descent is performed. After convergence of the optimization, the algorithm smooths the regions, and detects the edge and corner points. It assigns an edge probability to each point as a function of the curvature, and performs a region growing algorithm on an ϵ-neighborhood graph of the point cloud for regular points (Figure 2.19(c)). A second continuous optimization process is then performed, enforcing smoothness priors on edges. This step optimizes the position of edge, regular and corner points in order to locally maximize the model likelihood given the discrete attributes (Figure 2.19(d)).

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. they have been proved successful in surface reconstruction 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 an initial surface. A deformable model is a geometric object whose shape changes over time (Figure 2.20). The deformation behavior of a deformable model is governed by variational principles (VPs) and/or partial differential equations (PDEs).

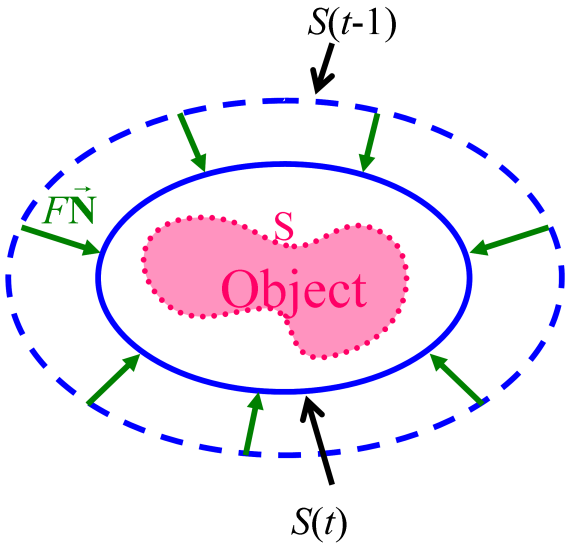


Figure 2.20: Active contours in 2D. The curve S is approximated dynamically by a family of curves $S(t), t = 0 \ldots n$. The curve $S(t)$ evolves to $S(t+1)$ according to $F\vec{N}$

The deformation algorithm is based on three components: the geometric representation (explicit, implicit or parametric), the evolution law and the topology control, e.g. the evolution may constraint the surface to have a genius of one.

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

$$\frac{\partial S}{\partial t} = F(S, N, K, f, ...), \tag{2.23}$$

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 $S(t=0) = S_0$ where 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 data 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. For example, Zhao et al. [ZOF01] proposed a 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 proportional to $-\nabla d(x) \cdot \vec{N} + T(x)$ where $\vec{N}$ is the inner normal at x and the tension of the surface represented by the second term $T(x)$ is not linear, so that the evolution process requires a large number of steps before reaching its equilibrium.

Chaine [Cha03], 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 a watertight surface maintained during evolution.

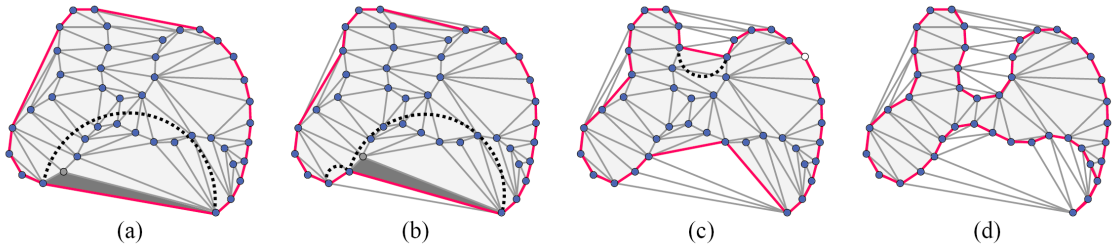


Figure 2.21: Geometric convection in 2D. From left to right: (a) Initialization of the curve by the convex hull of the point set. (b) The curve is updated. (c) An edge block a cavity. (d) the resulting curve.

At the beginning the surface is initialized as the convex hull of the point set (Figure 2.21(a)). The surface is then deformed using a set of geometric and topological operations (Figure 2.21(b)). The convection process corresponds to the first term of the evolution equation proposed by Zhao. A possible extension of the algorithm is proposed in which a curvature criterion is used corresponding to the tension term of Zhao evolution equation. Specifically, some *pocket* can appear as in Figure 2.21(c) and the algorithm must use other criteria based on the curvature to sculpt the pocket.

In [SLS+06], Sharf et al. present a deformable model that uses an explicit evolving front technique for reconstructing a 3D model. Their model includes multiple competing evolving fronts at different locations that converge towards the finer local features of the target shape only after reconstructing the coarse global features (Figure 2.22). The front evolution is guided by a scalar-field representing the distance from the point set (in outward normal direction while the initial surface is inside the point cloud). Thus, the front evolution is adapted to the local feature size of the shape.

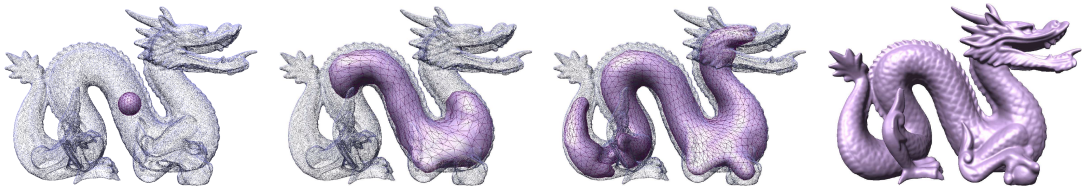


Figure 2.22: Reconstruction with a deformable model. From left to right (3 first images): Growing of a watertight (genus 0) mesh model inside the point cloud of the dragon. Right: the handle between the body and the tail is attached and the model is projected onto the point cloud.

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 [Kir00], 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 proved successful at reconstructing surfaces from point sets scattered on surfaces S of arbitrary topology. Furthermore, the reconstructed surface S' is watertight, holes are filled. Carr et al. in [CFB97] have proposed to fit skull surfaces using RBF algorithm. Usually skull defects are holes, the RBF reconstruction algorithm allows to create adapted cranial implant for cranioplasty. The RBF methods can be seen as variational problems [Duc77] which mathematical properties have been widely studied [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 :*

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

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

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

$\mathbb{R}$, $j = 1 \dots m$, thus f is written:

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

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

Minimizing the energy (3.2) consists then in solving a linear system with least-squares technique to determine the vector $\alpha = \{\alpha_1, \dots, \alpha_n\}$ by solving a linear system

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

where

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

3.1.1 Regularization Theory

When the reconstruction problem is ill-posed, ie. there is no solution or several solution, one approach is to add an *a priori* knowledge to the reconstruction process. A prior term, e.q., regularity information, may be added in the energy functional to constraint the solution to own some properties. The prior wishes on function regularity can be a surface with a minimum curvature or with a minimum area. A regularization term $\mathcal{E}_r(f)$ is added to the least squares term, $\mathcal{E}_s(f)$, in the energy functional E (3.2). Thus, the energy functional is defined as:

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

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 tuning the degree of smoothness. When $\lambda \to 0$, the surface closely fits the constraints. Conversely for largest values of λ, the surface becomes smoother and the accuracy of data point approximation decrease. (Figure 3.1).

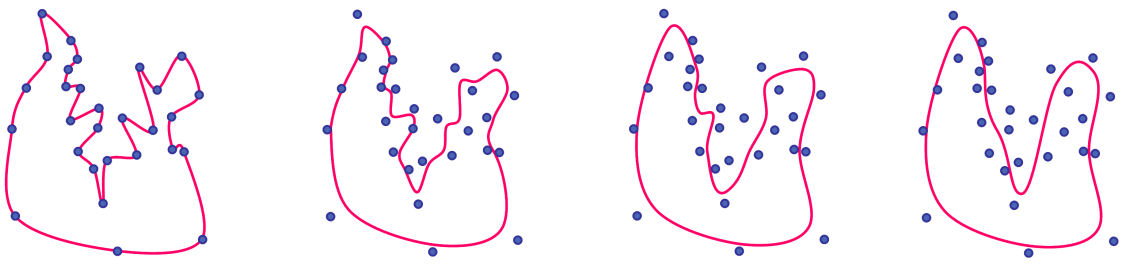


Figure 3.1: Regularization theory. Fitting a curve to a point set using an increasing value of λ.

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 being 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

For the interpolation problem, the function f must verify all constraints, i.e. the least square error must be zero, 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:

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

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 variables as in (5.9).

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

3.2 Radial Basis Functions

In the RBF approach, the basis functions are obtained from a single function $\phi : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$ 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 applications, we require the solution to be invariant by rigid transformation. We constrain the solution to be invariant through 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.

The RBF theory was initially developed for the interpolation problem case (Section 3.1.2) where the centers of the RBF are chosen to coincide with the constraints:

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

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

$$G_{X,\Phi} = \begin{pmatrix} \phi(\|x_1 - x_1\|) & \phi(\|x_1 - x_2\|) & \dots & \phi(\|x_1 - x_n\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|x_n - x_1\|) & \phi(\|x_n - x_2\|) & \dots & \phi(\|x_n - x_n\|) \end{pmatrix} \tag{3.15}$$

The main question is: how can we ensure that the interpolation matrix $G_{X,\Phi}$ is non singular?
Micchelli [Mic86] has shown that the distance matrix $G_{X,\Phi}$ generated by distinct points is invertible for several useful choices of ϕ. Comprehensive reviews were elaborated by Buhmann [Buh03] , Dyn [DR95] and Powell [Pow87]. 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 although 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 minimizes the curvature in 2D. The thin plate spline, in 2D, is the solution to a variational problem, the minimization of the integral:

$$\int_{\mathbb{R}^2} \left(\frac{\partial f}{\partial x_1 \partial x_1}\right)^2 + 2\left(\frac{\partial f}{\partial x_1 \partial x_2}\right)^2 + \left(\frac{\partial f}{\partial x_2 \partial x_2}\right)^2 \tag{3.16}$$

Remark: 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 the variational problem to the regularization theory (see Section 3.1.1) [Hay99].

Duchon has shown that seeking a function f such that f minimizes $\|D^2(f)\|$ and f verifies all the constraints is equivalent to interpolating these points using the biharmonic radial basis function $\phi(r) = r^2 log(|r|)$ (fig.3.2(e)). In 3D, 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 arbitrary dimension and that a class of basis functions was proposed in order to minimize (3.17)

$$\int_{\mathbb{R}^d} \sum_{|\alpha|=m} \frac{m!}{\alpha!} (D^\alpha f)^2 d\mathbb{R}^d. \tag{3.17}$$

For these basis function, the $G_{X,\Phi}$ naturally defines a quadratic form:

$$Q_{X,\phi} : (\alpha_1, \cdots, \alpha_n) \longmapsto \alpha^T G_{X,\phi} \alpha \tag{3.18}$$

on $\mathbb{R}^{\kappa}$ but these forms are positive definite only on a proper subspace of $\mathbb{R}^n$. In addition to reducing the α solution space, a polynomial reproduction condition may be required. Specifically, if the data come from a polynomial $q \in \mathbb{P}^d_m$, the interpolant must coincide with q, i.e $f(x) \equiv q(x)$ (see proofs in [Pow90, Buh03]). Thus a multivariate polynomial q is added to the weighted sum (see Equation (3.19)) in order to ensure positive definiteness of the solution, i.e. the space of admissible function is augmented by the space, $\mathbb{P}^d_m$, of polynomial of order up to m:

$$f(x) = \sum_{j=1}^{n} \alpha_j \phi(\|x - x_j\|) + q(x) \quad q \in \mathbb{P}^d_m. \tag{3.19}$$

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

$$f(x) = \sum_{j=1}^{n} \alpha_i \phi(\|x - x_j\|) + \sum_{k=1}^{l} \beta_k q_k(x). \tag{3.20}$$

Note that the interpolation problem (3.10) gives us n equations with $n + dim(\mathbb{P}^d_m)$ unknowns. 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*

$$\alpha^T G_{X,\phi} \alpha = \sum_{j,k=1}^{n} \alpha_j \alpha_k \phi(x_j, x_k) \tag{3.21}$$

of the quadric form (3.18) is positive, provided that the vector $\alpha = (\alpha_1, .., \alpha_n)$ has the additional property:

$$\sum_{j=1}^{n} \alpha_j q(x_j) = 0 \ \forall q \in \mathbb{P}^d_m, \tag{3.22}$$

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

Theorem 3.1. *Let ϕ be conditionally definite positive of order d. The function f defined in Equation (3.20) is the unique solution to the interpolation problem:*

$$\begin{cases} \sum_{j=1}^{n} \alpha_i \phi(\|x_i - x_j\|) + \sum_{k=1}^{l} \beta_k q_k(x_i) & \forall i = 1 \ldots n \\ \sum_{j=1}^{n} \alpha_j q_k(x_j) = 0 & \forall k = 0..l \end{cases} \tag{3.23}$$

Thus the interpolation system (3.23) can be expressed in matrix form as

$$\begin{pmatrix} G_{X,\phi} & Q_X \\ Q_X^t & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix} \tag{3.24}$$

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

$$\mathbf{Q_X} = \begin{pmatrix} q_1(x_1) & q_2(x_1) & \ldots & q_l(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ q_1(x_n) & q_2(x_n) & \ldots & q_l(x_n) \end{pmatrix} \tag{3.25}$$

Proof of Theorem 3.1: Let (α, β) be a pair of vectors that solve the homogeneous system with matrix (3.15). We have $G_{X,\phi}\alpha + Q\beta = F$ and $Q^t\alpha = 0$.
Multiplying the first equation 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 $\beta = 0$.
□

Examples of radial basis functions

- ≻ *Linear* (Figure 3.2(a)):

$$\phi(r) = |r| \tag{3.26}$$

  Note that the linear basis function corresponds, in one dimension, to the piecewise linear interpolation, that is the simplest case of spline interpolation.

- ≻ *Multiquadric* (Figure 3.2(b)):

$$\phi(r) = \sqrt{r^2 + \rho^2} \qquad \rho \in \mathbb{R} \tag{3.27}$$

- ≻ *Inverse multiquadric* (Figure 3.2(d)):

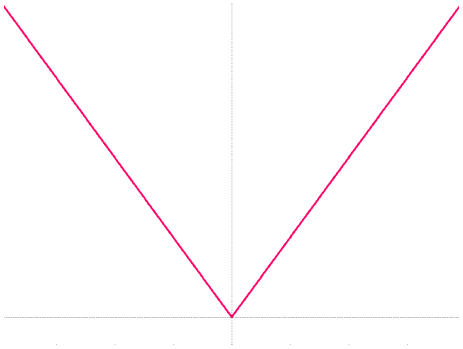
$$\phi(r) = \frac{1}{\sqrt{r^2 + \rho^2}} \qquad \rho \in \mathbb{R} \tag{3.28}$$

- ≻ *Gaussian* function (Figure 3.2(c)):

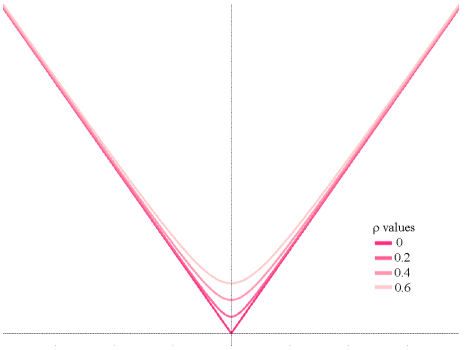
$$\phi(r) = e^{-\rho * r^2} \qquad \rho \in \mathbb{R}, \tag{3.29}$$

  (the common basis function in neural network).

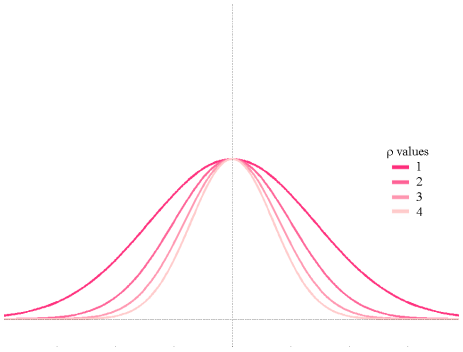
The inverse multiquadrics and the Gaussian function share a common property: they are both localized functions, i.e. with bounded values (Figure 3.2(c) and Figure 3.2(d)).



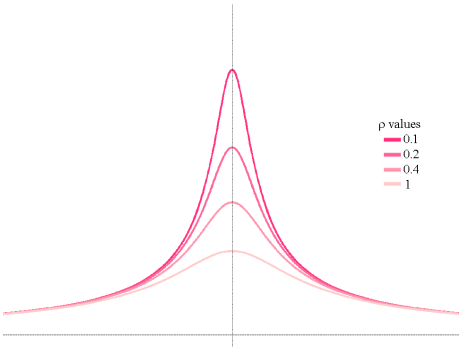
(a) Linear function:$\phi(r) = |r|$



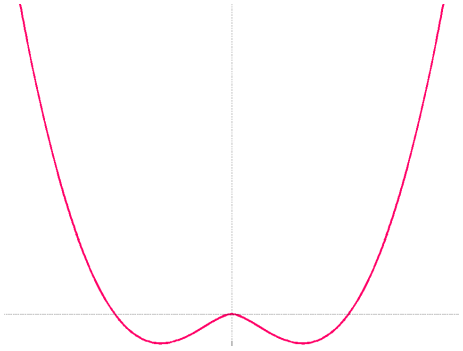
(b) Multiquadric: $\phi(r) = \sqrt{r^2 + \rho^2}$



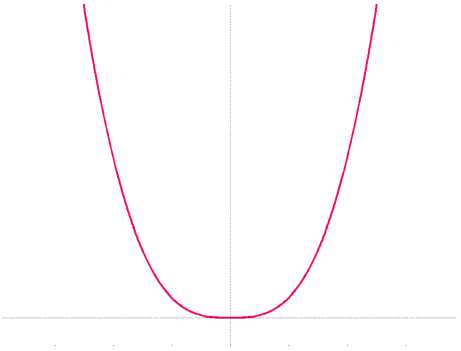
(c) Gaussian function $\phi(r) = e^{-\rho * r^2}$



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



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

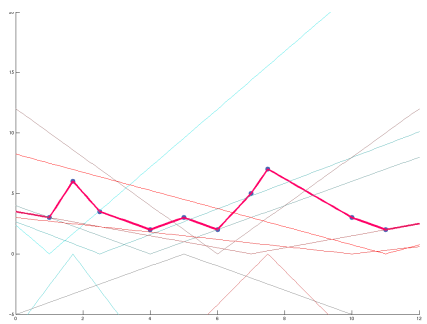


(f) Triharmonic: $\phi(r) = r^3$

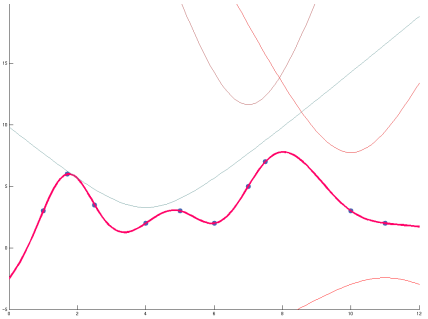
Figure 3.2: Different classes of RBF

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

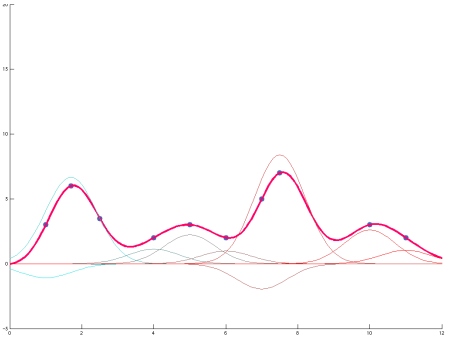
Figure 3.3 shows different reconstructions of the same point set using the different classes of RBF described above.



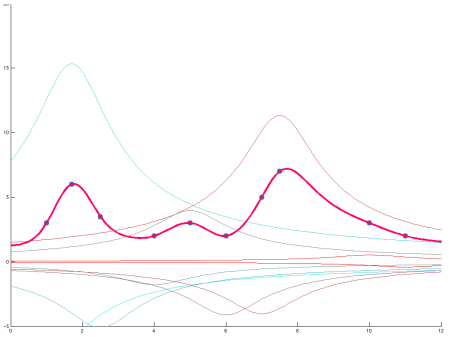
(a) Linear function



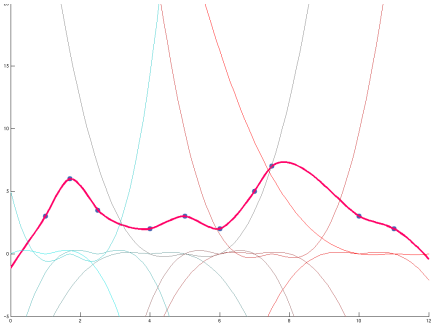
(b) Multiquadric



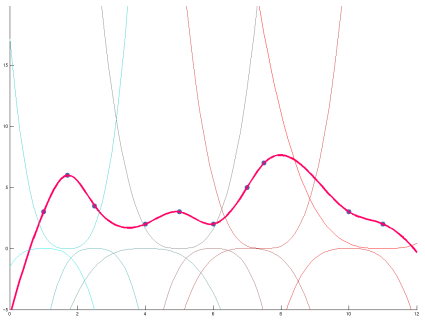
(c) Gaussian



(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 point is reconstructed. The resulting curve is constructed as a mixture of the basis functions. The curves of the functions $r \mapsto \alpha_j \phi(r, c_j)$ are plotted.

3.3 RBF Surface reconstruction

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

$$S' = \left\{ x \in \mathbb{R}^3 : f(x) = 0 \right\} \tag{3.30}$$

where $f(x)$ is expressed as a weighted sum of basis functions ϕ centered at some points c_j called centers. Thus Equation (3.3) become:

$$f(x) = \sum_{j=0}^{n} \alpha_j \phi(\|x - c_j\|) \tag{3.31}$$

The input is $P = \{p_i\}_{i=1...n}$ 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 to n scalar values $\{f_i\}$ such that $f_i = 0$.

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

$$E(f) = \sum_{i=1}^{n} \left(f_i - \sum_{j=0}^{m} \alpha_j \phi(\|p_i - c_j\|) \right)^2 . \tag{3.32}$$

To avoid the trivial solution $\alpha = \overrightarrow{0}$, several interior and exterior constraints are added where the function f does not vanish (Figure 3.3.1). The additional constraints are called *off-constraints*. Usually, at each input point p_i is associated two off-constraint points: each

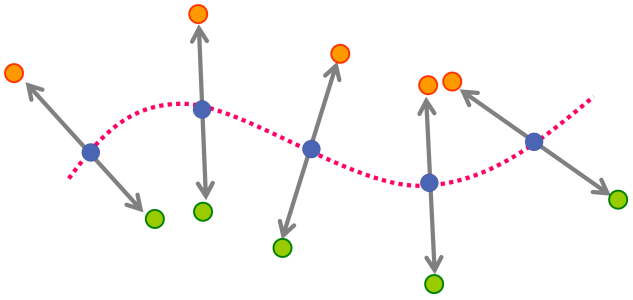


Figure 3.4: Additional off-surface points along the surface normals.

off-constraint point lie on the normal at p_i on each side of the inferred surface. Note that an oriented normal must be known. For each off-constraint $q_k \in Q$, we assign to f a signed value $f_k = \pm d(q_k, P)$, where $d(q_k, P)$ is the distance from q_k to the point set P. The N constraints $X = \{x_i\}_{i=1...N}$ 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 $N \approx 3n$. Note that the normals to the surface S need to be known in order to compute the off-constraints.

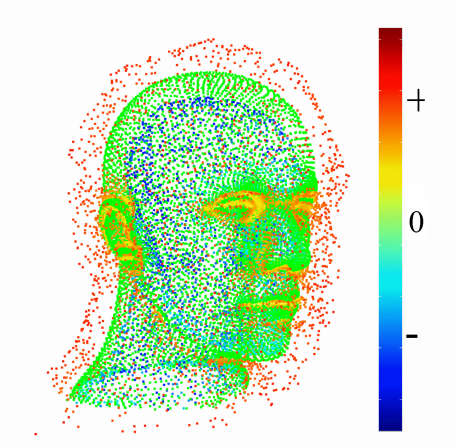


Figure 3.5: Constraints points. The colors represent the function values (cold tones for negative, hot tones for positive and green for zero values).

3.3.2 Centers

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

$$c_i = x_i, \qquad \forall i = 1 \dots N \tag{3.33}$$

The minimization process (3.1) reduces to solving a $N \times N$ linear system which requires $O(N^3)$ machine operations and $O(N^2)$ 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, a first idea is to reduce the number of constraints. Note that since most algorithms use the same points both as constraints and centers, this also leads to center reduction. This approach is commonly called *center reduction* in the literature.

Center Reduction Some constraint points are relevant while some others are redundant or simply irrelevant. Therefore dealing with fewer constraints points is useful.
Constraint reduction consists in approximating, with the desired accuracy, all input data using fewer constraints but a sufficient subset of constraints. If the function f is defined to be centered at the constraint points, then reducing the constraints is equivalent to 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 at all input points.

3.3.3 Basis functions

All functions listed in Section 3.2 have an unbounded support. The corresponding equations lead to a dense linear system. Therefore recovering a solution is tractable only for

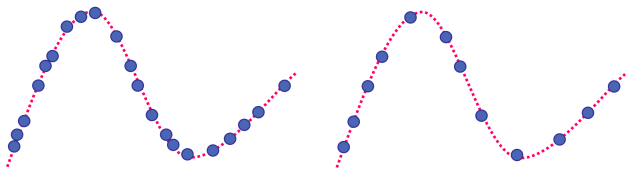


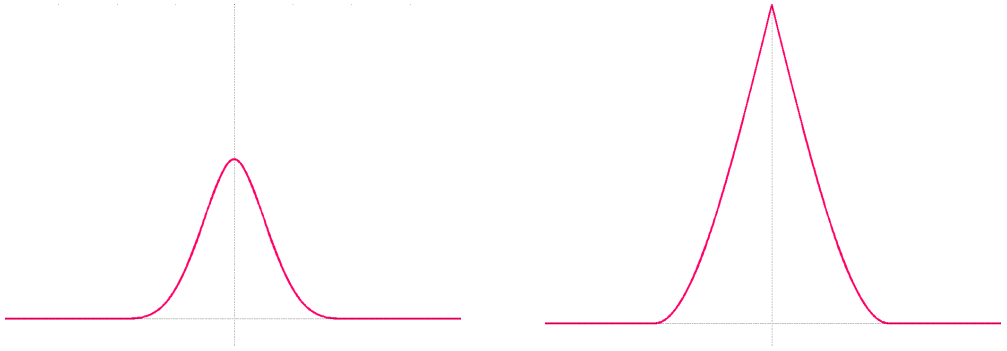
Figure 3.6: Center reduction as performed by [CBC+01].

small data sets. In order to obtain a sparse interpolation matrix, Compactly Supported RBFs (CSRBF) have been introduced by Wendland in [Wen95]. Other CSRBFs were proposed in the literature [Sch95, Wu95]. Note that these basis functions are not suited to reconstruction from inhomogeneously sampled surfaces.

Two examples of CSRBF (Figure 3.7) are:

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

where $(x)_+ = x$ if $x > 0$ and $(x)_+ = 0$ otherwise.



(a) Wendland function :$\phi(r) = (1-r)_+^4(1+4r)$

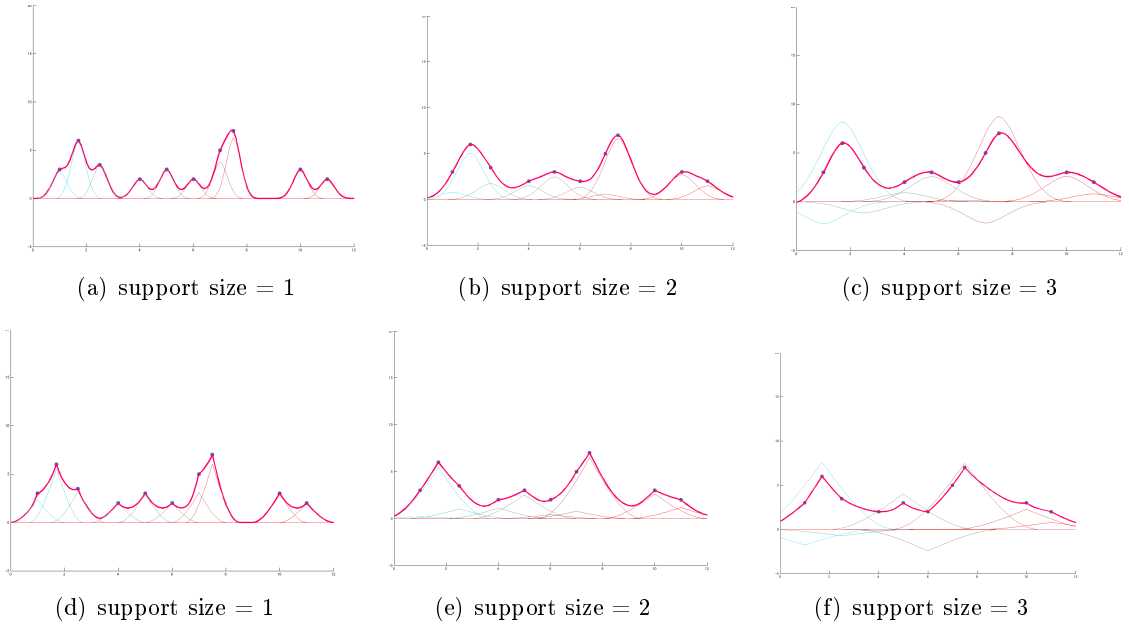
(b) Wu function : $\phi(r) = (1-r)_+^2(2+r)$

Figure 3.7: Two compactly supported RBF.

The size of the support is important. Beside acting on the sparsity of the matrix, it infers on the property of the reconstructed function. The larger the support, the smoother the function, as shown in Figure 3.8, but the denser is the matrix. Typically, the support size is the same for all the basis functions.

Ohtake et al. [OBS04] have proposed to locally adapt the support of the basis functions. A support σ_j is associated to 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 smaller than a certain threshold.

We can note that radially symmetric functions are not suited to piecewise smooth surface reconstruction. Dinh et al. [DTS01] have proposed to overcome this issue by using anisotropic basis functions. However, as the basis function are smooth, the function solution remaining smooth too even in case of extreme anisotropy.



(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 non uniform 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. The curves of the functions $r \mapsto \alpha_j \phi(r, c_j)$ are plotted.

3.3.4 In Practice

The RBF approach can be summarized by solving a linear system. However, in practice, the system may be very large. The naive approach requires $O(N^3)$ machine operations to solve the system and $0(N^2)$ bits for storage. Several important strategies were proposed: data reduction, for example the *center reduction* (Section 3.3.2), or sparsening the matrix with compactly supported RBF (Section 3.3.3). In the following, we present additional 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 be well suited. Thus polyharmonic basis functions must be used. In the case of globally supported basis functions, like polyharmonic RBF, m evaluation of $f(x)$ require $O(mN)$ operations, where N is the number of constraints.

In [CBC+01] Carr and Beatson have proposed to perform fast evaluation using the Fast Multipole Method (FMM). This algorithm reduces 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.

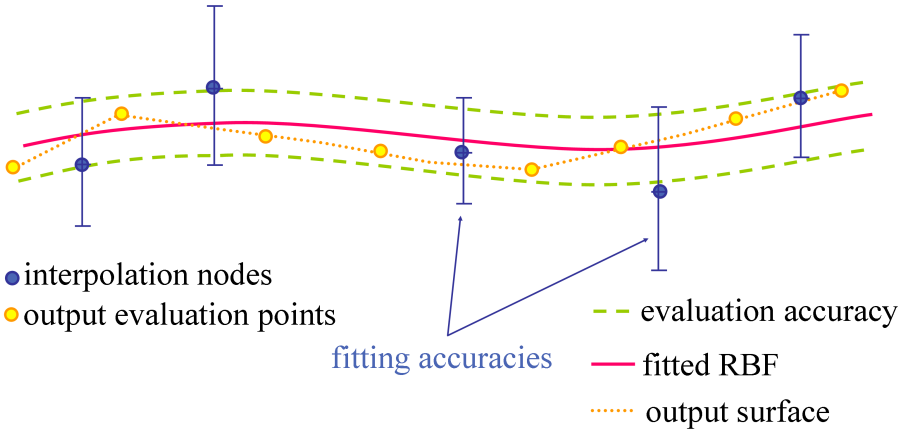


Figure 3.9: Fast evaluation using FMM. A RBF is evaluated at regular intervals lying between the dashed evaluation accuracy bands either side of the actual function.

To perform the approximations two parameters are introduced: a fitting accuracy and an evaluation accuracy. The fitting accuracy specifies the maximum allowed deviation between 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.

Note that implementing a FMM is notoriously difficult.

3.3.4.2 Domain Decomposition and Multilevel Methods

The original data set is subdivided into several smaller data sets. The problem is then solved iteratively for each cluster. In addition to yield piecewise smooth surface reconstruction, domain decomposition approaches allow for dealing with very large point sets and performing local reconstruction. Finally, the underlying data structure provides us with a multiscale representation.

Partition of Unity Wendland, in a theoretical survey [Wen02], combine the Partition of Unity (see sec.2.2.2) method with 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 "slightly" overlapping subdomains $\{\Omega_i\}_{i=1...M}$ such as $\Omega \subset \cup_i \Omega_i$ (Figure 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 f is constructed by blending together all the local functions f_i, f is thus defined as

$$f(x) = \sum_{i=1}^{M} f_i(x) w_i(x) \tag{3.34}$$

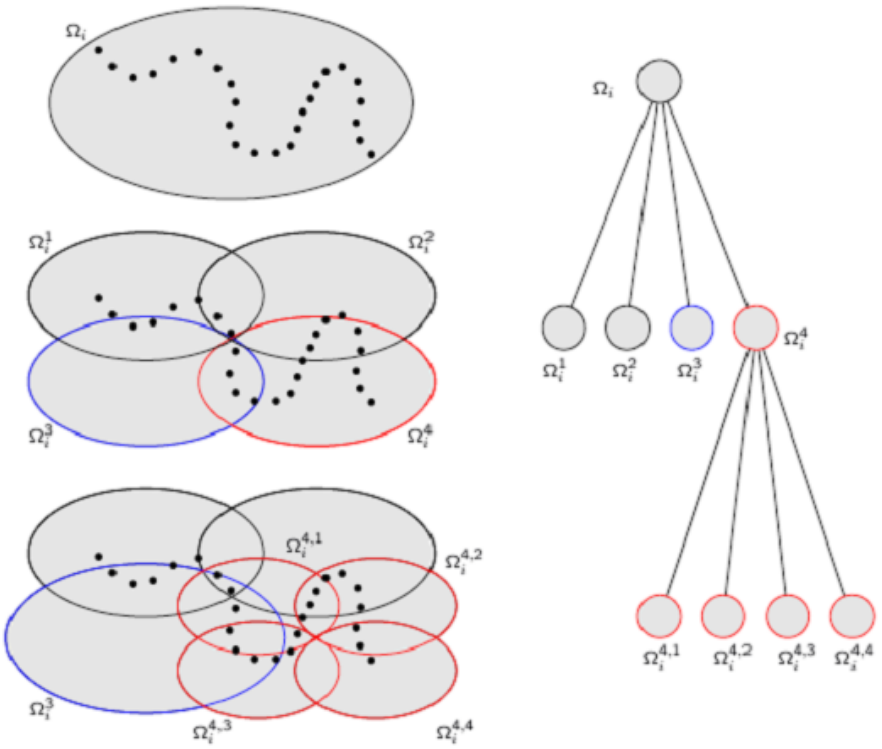


Figure 3.10: Partition of unity. Left: The space is subdivided according to the number of points in each cell. Right: the tree corresponding to the different levels whose the leaves are the final subdomains.

where the w_i are weighting functions. In order to perform a partition of unity algorithm, the weights 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:

$$w_i(x) = \frac{W_i(x)}{\sum_{j=1}^{M} W_j(x)} \quad \forall i = 1 \dots M \tag{3.35}$$

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, OBS05] 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 used to recover the small details.

The function solution f is formulated as

$$f(x) = \sum_{c_i} [g_i(x) + \lambda_i] \phi_{\sigma_i}(\|x - c_i\|) \tag{3.36}$$

where the unknowns are the functions g_i and the weights λ_i. Each 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:

$$f(c_i) = 0 \tag{3.37}$$

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

$$\phi_{\sigma_i}(x) = \phi(\frac{x}{\sigma_i}) \quad \forall i = 1 \dots n \tag{3.38}$$

The equation (3.36) can be rewritten as

$$f(x) = \sum_{p_i} g_i(x)\phi_{\sigma_i}(\|x - c_i\|) + \sum_{p_i} \lambda_i \phi_{\sigma_i}(\|x - c_i\|) \tag{3.39}$$

The first term of the right-hand side of (3.39) can be considered as a base approximation and the second term represents local details.

The centers c_i and their area of influence 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. To 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 (3.36) at the level k is computed according to the function computed at the level $k-1$:

$$f^k(x) = f^{k-1}(x) + o^k(x) \tag{3.40}$$

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

$$o^k(x) = \sum_{p_i^k \in \mathcal{P}^k} [g_i^k(x) + \lambda_i^k] \phi_{\sigma_i}^k(\|x - p_i^k\|) \tag{3.41}$$

$\mathcal{P}^k$ is the set of centers at the level k approximated by f^k . In the first level, $\mathcal{P}^1$ corresponds

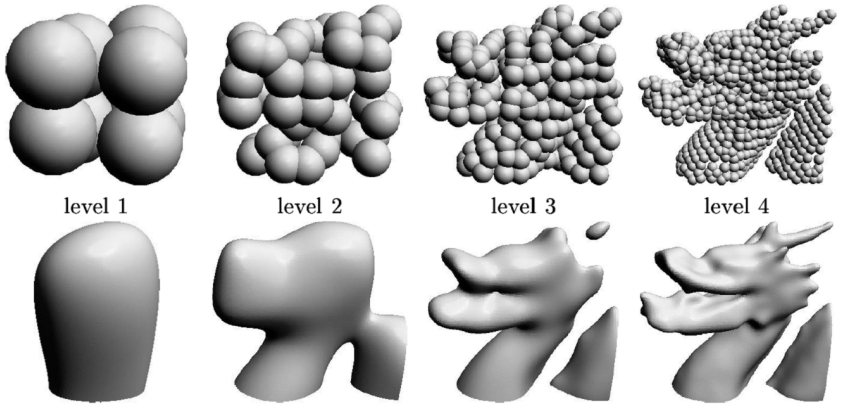


Figure 3.11: Multi-scale interpolation of the Stanford dragon model. From left to right: four first levels of the multi-scale hierarchy. Top row: the spheres corresponding to the support of the RBF. Bottom row: the zero level set of the interpolating function.

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 "little" 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:

$$f(x) = \sum_{j=1}^{m} \alpha_j \phi(\|x - c_j\|), \tag{3.42}$$

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

$$G_{X,C,\phi} = \begin{pmatrix} \phi(\|x_1 - c_1\|) & \phi(\|x - 1 - c_2\|) & \dots & \phi(\|x_1 - c_m\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|x_N - c_1\|)) & \phi(\|x_N - c_2\|) & \dots & \phi(\|x_N - c_m\|) \end{pmatrix} \tag{3.43}$$

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 many constraints as we want, i.e as much desired information. 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 involves in 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 our work, we exploit one of the most important degrees of freedom offered: the location of centers and constraints to obtain a satisfactory trade-off between number of centers and fitting accuracy. In addition to this choice, we propose to use compactly supported RBF whose support is adapted to the local geometry of the point set P. Our aim is to obtain a compact representation and a reconstruction algorithm with few parameters.

Part II

Contribution: Voronoi Centered Radial Basis Functions

Chapter 4

Introduction

Although Voronoi-based reconstruction has long been criticized for its computational complexity, recent developments in the implementation of fast algorithms have alleviated this issue. As an example, computing the Delaunay triangulation of 50K points takes 1s using the CGAL library [FGK+00]. The efficiency and the accuracy of 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 with few exceptions such as tight cocone (see Section 2.1.4).

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 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 to improve 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 specifically, 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 reconstruction process to accept noisy data and to adapt the level of detail to the allocated budget of centers.

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 reconstruction 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 is the following: given a point set $P = \{p_i\}_{i=1..n} \subset \mathbb{R}^3$ measured on a surface S, we want to construct a surface S' that approximates S. 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 functions. In our algorithm, we explore the main degrees of freedom of the generalized radial basis function approach: the number of centers and constraints and their location as well as the type of basis functions.

The outline of this part is as follow. Chapter 5 contains an overview of our algorithm. In Chapter 6, we detail the different strategies for selecting centers. We then provide in Section 7 the implementation details and optimization of the constraint classification and of the matrix construction. Note that Section 7 is independent from Chapter 8. Finally, Chapter 8 contains several results which illustrate 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.
Recall here that we restrict our work to surface without boundaries where the surface divides the space into two parts: 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, defined as a weighted sum of radial basis functions:

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

where $\{c_j\}_{j=1...m}$ is the set of centers, ϕ a basis function and $\{\alpha_j\}_{j=1...m}$ is the set of unknown weights.

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:

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

Our aim is to find the vector α 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 RBFs (see Section 3.4) where the functions ϕ are centered at a subset of Voronoi vertices. Our reconstruction adapts to the number of centers, m, specified by the user.

Our algorithm proceeds as follows: we first compute the Delaunay triangulation of the input points (as well as its dual Voronoi diagram). Our algorithm then defines a selection 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 so

as to sample a part of the medial axis. This selection is performed either by filtering and clustering the set of poles, or by selecting the poles with a greedy algorithm.

In the first selection approach, 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 greedy selection approach, the m poles are selected in order to achieve a nice sampling of the medial axis which simultaneously adjusts the level of detail to the allocated 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. The 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 fitting of the function values f_i at data points p_i and at 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 for.

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 that the vertices of the cell which are the furthest away from p on the two sides of the inferred surface are called poles of p. Each pole is the center of a Delaunay ball called *polar ball* (see Section 2.1.2). A polar ball is a maximal ball empty of sample points. Such a ball is close to a maximal ball in $\mathbb{R}^3 \setminus S$.

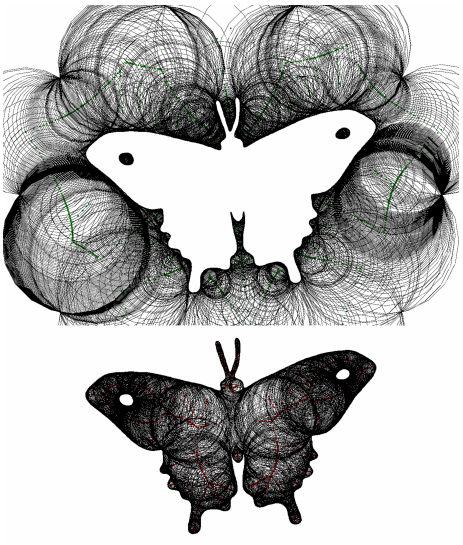
The rational behind our idea is that a solid can be roughly approximated (exact in the limit) as a union of balls: the Medial Axis Transform.

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

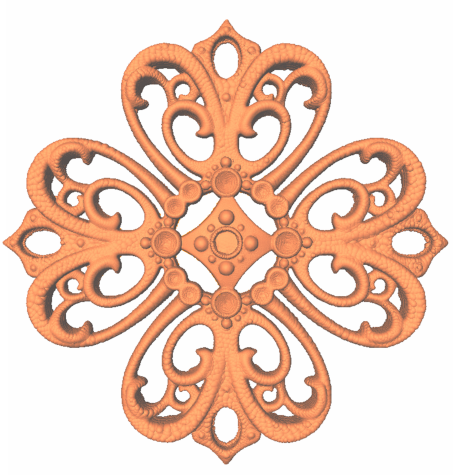
Considering that any smooth surface S can be viewed as the envelope of the maximal balls in $\mathbb{R}^3 \setminus S$, using poles as centers for radial basis functions is a rather natural idea.

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 (Figure 5.2);
- only a discrete approximation of the medial axis is known and its sampling is dependent of the distribution of the input points.



(a) MAT in 2D: outside (top) and inside (bottom) component.



(b) MAT 3D: only the inside component is shown.

Figure 5.1: Medial Axis Transform (MAT)

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 little hope to reconstruct thin details and thus we need to remove the poles which correspond to the smallest details. Note that small details are in general hardly distinguishable from noise. Furthermore, our sampling density must be independent from the distribution of the data points, thus we propose to perform a sampling according to a sizing function.

The *sizing function*, *sf*, is defined on the medial axis, and 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 continuous on each component of the medial axis. Chapter 6 details the selection of the m centers.

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.

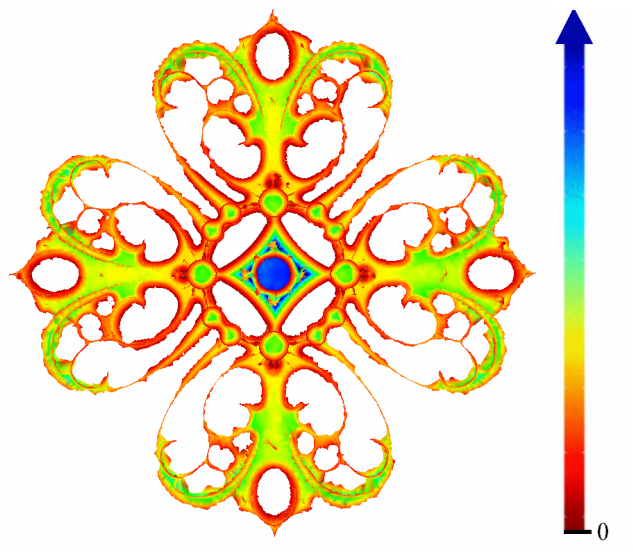


Figure 5.3: Sizing function on the medial axis in 3D. The colors represent the sizing values; warm tones for the minimum and cold tones for the maximal values)

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 (Figure 5.4).

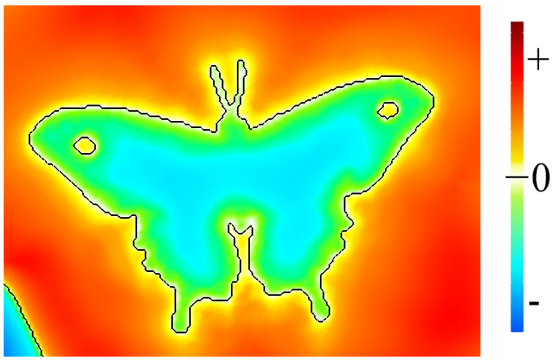


Figure 5.4: 2D shape (black) and the computed implicit function. Colors range from cold color tones for positive distance values to hot color tones for negative distance values.

As discussed in 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 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 faster and more robust against noise. During the classification process, a location tag (inside, outside or 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 as a constraint. The pole classification algorithm is detailed in Section 7.1.

5.3 Basis Functions

Recall that the reconstruction process is required to be invariant under any Euclidean transformation. The function Φ is thus restricted to the set of radial functions (see Section 3.2):

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

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

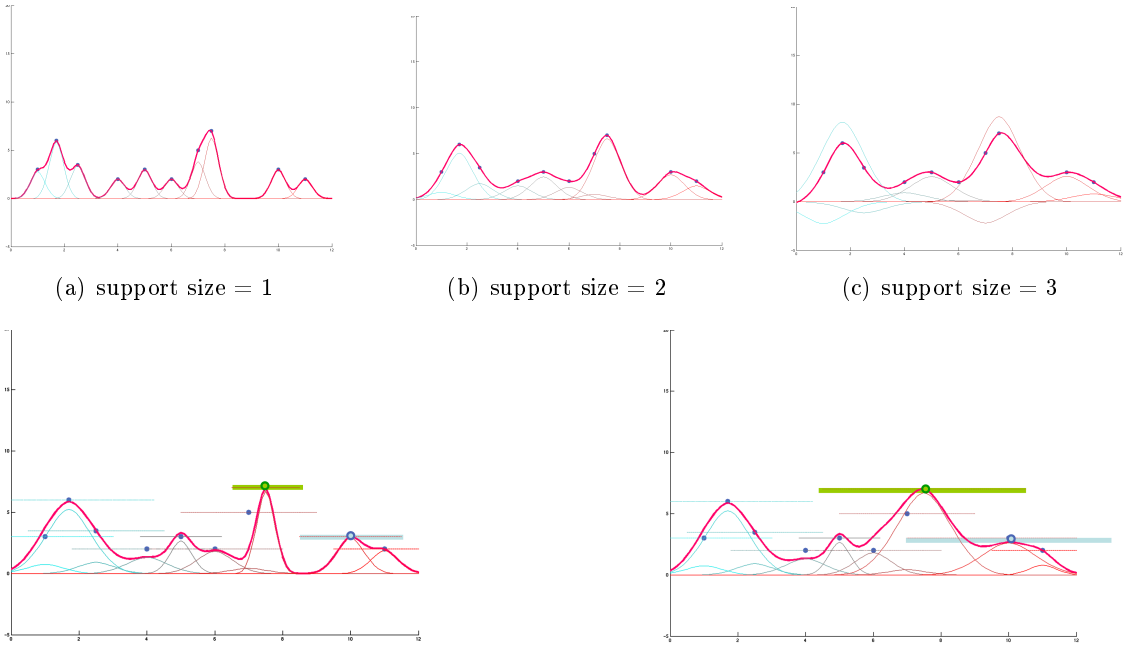
We choose a Gaussian-like function with a compact support [Wen95] whose size is locally adapted.
As shown in Section 3.3.3, the support size of the basis function impacts the reconstruction result (Figure 5.5). More specifically, the support size allows for adjusting the reconstruction to the geometry of the curve in term of continuity, connexity.

In the examples shown in Figure 5.5, the support is global and thus it is difficult to adapt the reconstruction to the local shape geometry. Different local supports can be used to handle this problem as shown by Figure 5.5(d).

As centers are poles, to each center c_i corresponds 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 at c_i is computed from r_i. Moreover, the ϕ function (5.4) centered at c_i is scaled according to the local support s_i:

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

We want to use a class of radial basis functions ϕ_j compactly supported in order to



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

(d) non uniform support size (e) non uniform support size

Figure 5.5: The set of blue points is reconstructed using the Wendland functions (red curve).
(a),(b) and (c): Uniform support sizes.
(d) and (e): Non uniform support size fixed arbitrary. On (e) the support size of the function centered at the blue point and at the green point increase.
The curves of the functions $r \mapsto \alpha_j \phi(r, c_j)$ are plotted.

obtain a sparse matrix and hence to perform efficient evaluations. Furthermore, we want a class of smooth basis functions since the basis function determines the smoothness of the approximant. A family of piecewise polynomial functions with local support was defined as:

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

where q denotes a univariate polynomial. Wendland [Wen95] have proposed several basis functions of minimal degree. In our implementation we use the following function:

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

where $(x)_+ = x$ if $x > 0$ and $(x)_+ = 0$ otherwise. This function is smooth (C^2) and Gaussian-like (see Figure 5.5).

5.4 System Solving

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

n input points and the additional off-surface points. To solve the reconstruction problem, the following energy is minimized:

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

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 the linear system:

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

where G is the matrix $[\phi_j(\|q_i - c_j\|)]_{i=1..N, j=1..m}$ and F be the vector $[f_i]_{i=1..N}$. In the following, the matrix G is defined as:

$$\mathbf{G} = \begin{pmatrix} \phi_1(\|q_1 - c_1\|) & \ldots & \phi_m(\|q_1 - c_m\|) \\ \vdots & \vdots & \ddots \\ \phi_1(\|q_N - c_1\|) & \ldots & \phi_m(\|q_N - c_m\|) \end{pmatrix} \tag{5.10}$$

An approximation using the least squares method implies solving the system (5.9). With the simpler 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. Different constructions of Matrix $G^t G$ are detailed in Section 7.2.

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. Specifically, 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.

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 at these points is an approximation of S.

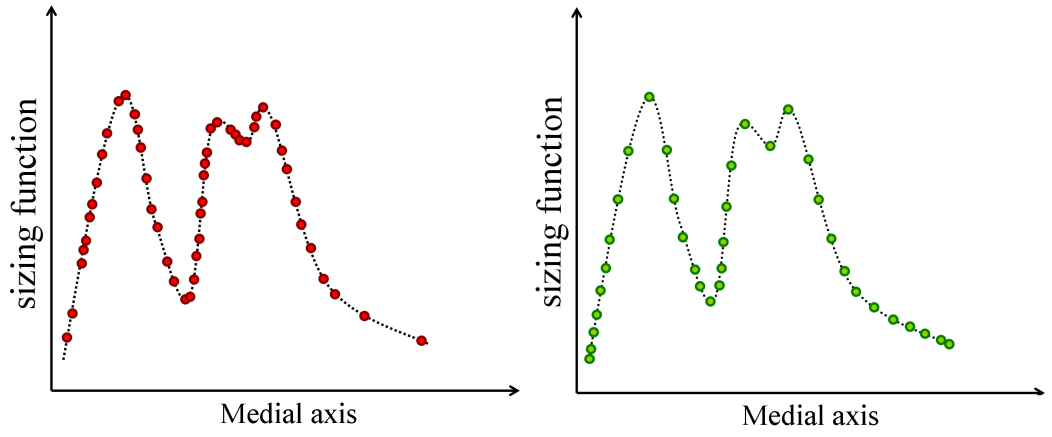


Figure 6.1: Sizing function on the medial axis in 2D. Left (red): at each point corresponds a pole. Right (green): after resampling, the sampling density is locally adapted to the sizing function.

The surface S being unknown, thus we do not have a continuous representation of the medial axis of S. However, we can approximate the medial axis by selecting a subset of the Voronoi vertices (see Section 2.1.2), the set of poles which approximates the medial axis. In addition, we can define the sizing function (see 70) on the medial axis by using the radius values of the polar balls centered at the poles.

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 upon the notion of λ-medial axis. The filtering step allows for removing the small features or the noise while 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 discard the poles whose the maximal balls intersect deeply the maximal ball of v. The distribution of poles on the approximate medial axis is

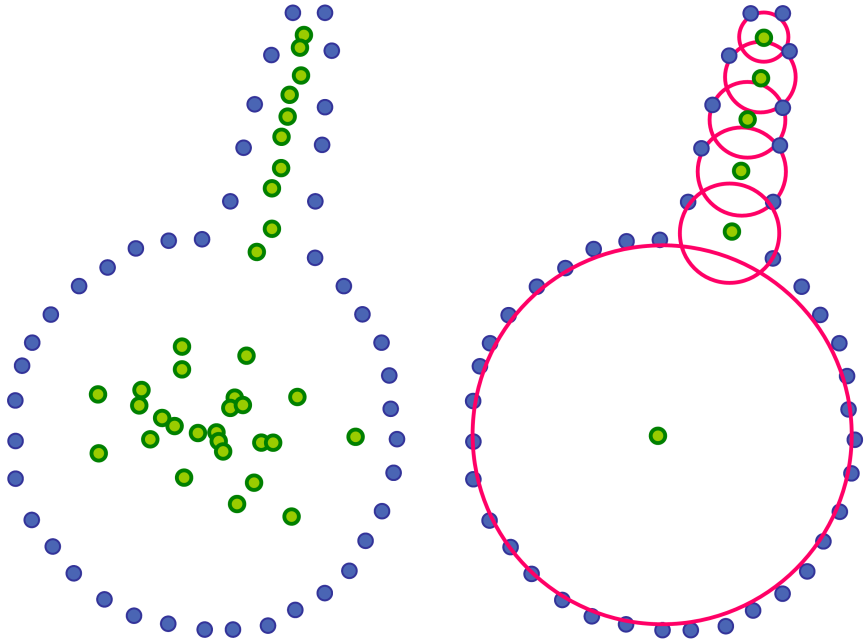


Figure 6.2: Distribution of the polar balls. A point set (blue) measured on a circle with a bump and the set of inside poles (green). Left: The distribution of poles is highly dependent on the distribution of the input points. Right: 6 centers (green) are selected, their polar balls (pink) produce a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

highly dependent on the distribution of the input points, see Figure 6.1. The distribution of polar balls centered at the centers needs to be relevant, i.e. to satisfy a minimum of overlapping (Figure 6.2). Thus, a re sampling of the medial axis is necessary to adapt the sampling density to the local geometry and to the desired number of centers.

Besides the local geometry, the sampling must be adapted to the number of centers m. If m is small, there is little hope to reconstruct small details. Therefore, we need to discard the poles which correspond to the smallest details, which are hardly distinguishable from noise. A filtering of the poles, according to their polar balls, must be performed in order to adapt the sampling to the level of detail fixed by the allocated number of centers (as shown in Figure 6.2).

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 pole *filtering* (Section 6.1.1) followed by a *clustering* (Section 6.1.2). The pole filtering is based on the notion of 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* (Section 6.2) of m poles. The procedure select the poles ordered by decreasing polar ball radius, while disqualifying all unselected poles whose support deeply intersects the poles just selected. The selection stops when m poles have been selected to be the centers. We obtain a center distribution adapted to the geometry of the shape. In addition, the level of detail is adapted to the allocated center budget m, while poles corresponding to the smallest polar balls are not selected if m is too small.

6.1 Filtering and Clustering

6.1.1 Medial Axis Filtering

A problem arises in 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. Even if two shapes are very close, e.g in terms of their Hausdorff distance, they may have very different medial axes (see Figure 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 [AMTI96, 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 of points on the surface S that are closest to x.

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

In general the cardinality of this set denoted $|\Gamma(x)|$ is 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, the real-valued function $\gamma(x)$ is defined as the radius of the smallest enclosing ball of the $\Gamma(x)$. 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.

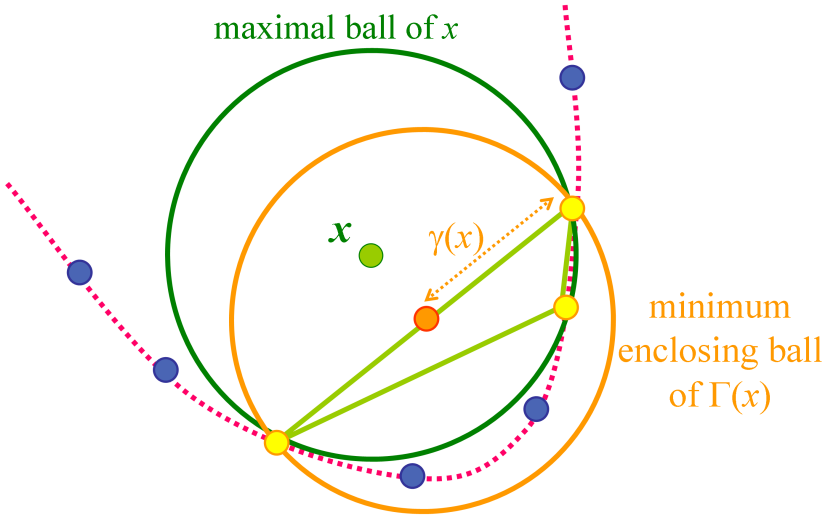


Figure 6.3: λ-medial axis criterion: the radius of the minimum enclosing ball (orange) has to be greater than λ. Relative λ-medial axis: the ratio of the radii of the two balls must be greater than $\lambda_r \in [0 \dots 1]$.

Angle and Distance Removing Criterion Attali and Montanvert [AMTI96, AM97] proposed a scale-invariant "removing" criterion based on a *bisector angle* and the notion of *thickness* (Figure 6.4). Let v be a point on the medial axis. The thickness is defined by

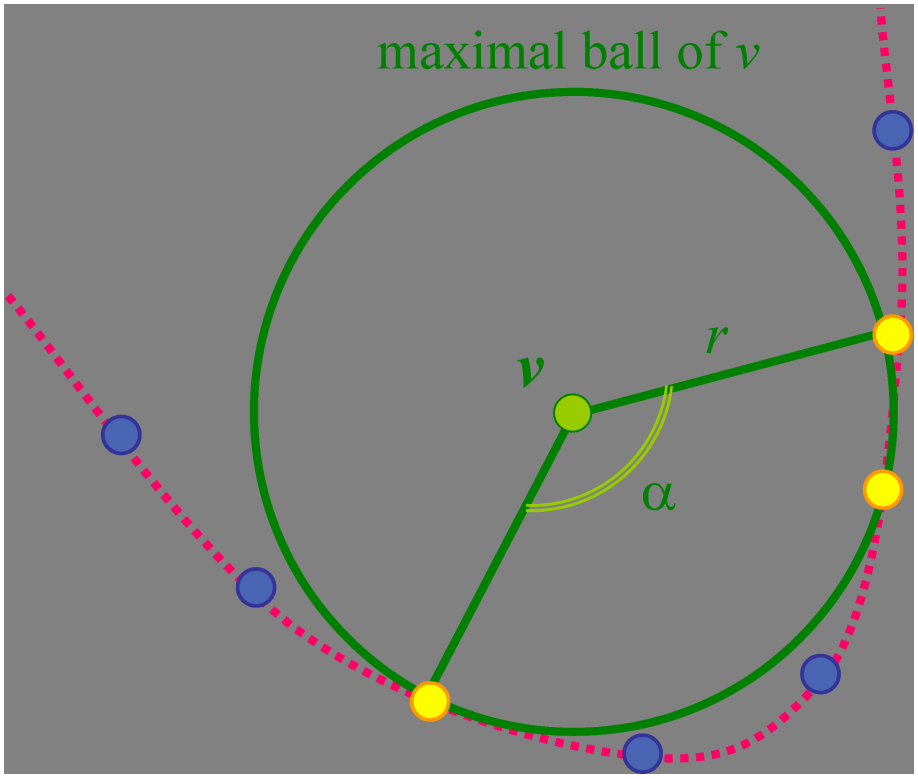


Figure 6.4: Bisector angle $\alpha(v)$ and thickness $\rho(v)$ on a 2D shape.

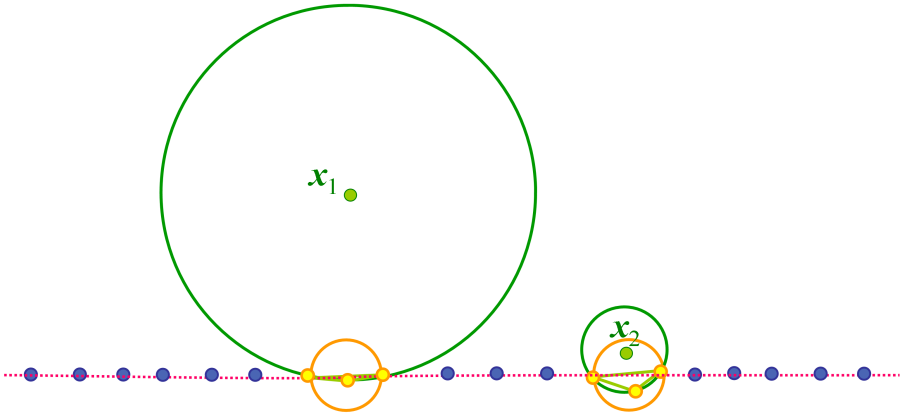


Figure 6.5: Poles to be removed by filtering.

the radius $r(v)$ of the maximal ball centered at the point v. The bisector angle is defined by the maximal angle $\alpha(v)$ between v and two of the contact points where the polar ball centered at v touches the surface S. points of the medial axis associated to noisy input points are characterized by a small bisector angle and a small thickness. Therefore two thresholds are proposed: α_0 and r_0 and the simplification consists in keeping the skeleton points v such that

$$\alpha(v) > \alpha_0 \text{ or } r(v) > r_0. \tag{6.3}$$

The relation between λ, α and r is given by:

$$\lambda(v) = r(v) \times \sin\left(\frac{\alpha(v)}{2}\right). \tag{6.4}$$

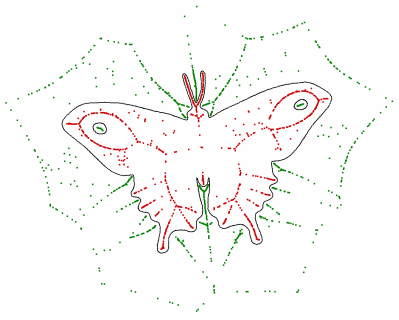
Relative λ Criterion The relation (6.4) leads to another criterion to filter the medial axis, the *relative* λ. The filtering is performed with the ratio between $r(v)$ and $\gamma(v)$, that is the ratio between the radius of the minimum enclosing ball and the radius of the maximal ball (Figure 6.3).

Figure 6.5 illustrates this interest of this criterion. With a λ-medial axis filtering the two poles, x_1 and x_2 are removed since the radius of their minimum enclosing balls are the same, i.e $\gamma(x_1) = \gamma(x_2)$. With the ratio criterion, the pole x_1 is removed whereas x_2 is not removed since $\frac{\gamma(x_2)}{r(x_2)} >> \frac{\gamma(x_1)}{r(x_1)}$.

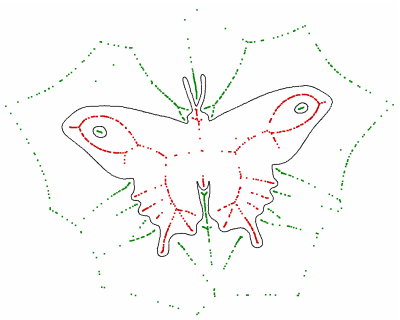
Our implementation We use the idea of λ-medial axis to perform the pole filtering in order to smooth noise and to adapt the level of detail of the reconstruction to the allocated budget of centers. Specifically, this implies that we determine the value λ suitable to the sampled shape and to the budget of centers, and that we 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 (Figure 6.6(a) and Figure 6.6(b)). Note that if the λ value is too large, some details are missed such as the antenna of the

butterfly for λ=0.01 (see Figure 6.6(b)).

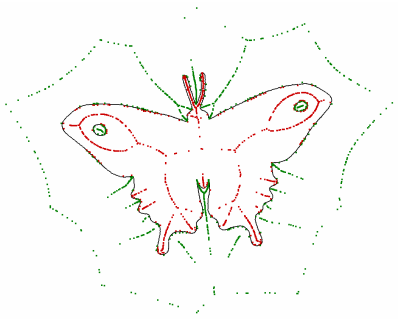
Figure 6.6 shows examples of filtering. We can note that in general we obtain the best



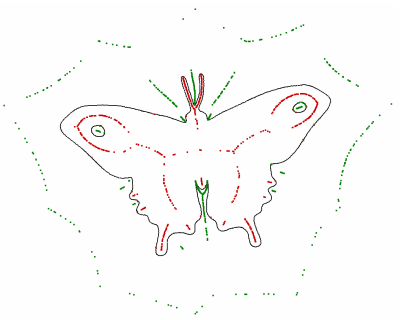
(a) λ-medial axis, $\lambda = 0.005$



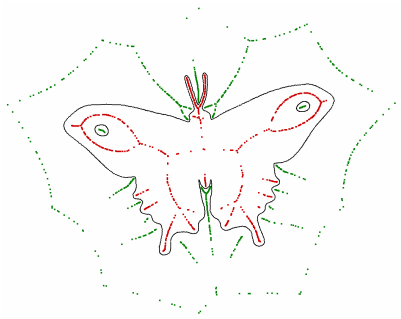
(b) λ-medial axis, $\lambda = 0.01$



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



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



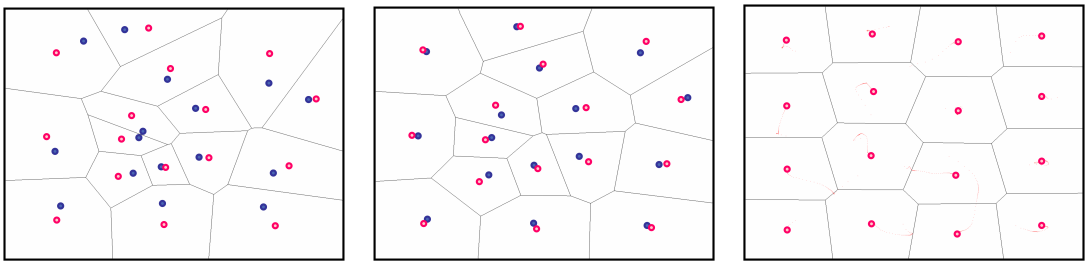
(e) Combined filtering, λ= 0.005 and $\lambda_r = 0.5$

Figure 6.6: Medial axis filtering with different λ criteria and relative λ_r criteria. Poles after medial axis filtering are depicted (red for inside poles and green for outside poles). To get a better sense of the λ parameter: the diagonal length of the bounding box of the input point set is 1.4.

result by combining the two criteria (relative and note relative).

6.1.2 Pole Clustering

Let m be the desired number of centers, 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 most relevant generators from PC, where *relevant* means a set of m points which is



(a) (b) (c)

Figure 6.7: Two dimensional example: Lloyd algorithm with a uniform density.
(a)initial generator set (blue), the Voronoi diagram of the generator and the centroid (red);
(b)After one iteration: the new centroid and the new clusters
(c)After convergence: the final centroids, the final clusters and the centroid trajectories during the algorithm.

a subsampling of PC with several properties like a desired distribution.
Remarks: the set PC forms a good approximation of the filtered medial axis of the surface S, but its sampling is highly dependent on 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 centers [Mac67].
Remind that our goal is to obtain a sampling of the λ-medial axis according to the local sizing field function, *sf*. Specifically, in the center set, the distance between a center and its nearest neighbor needs to be proportional to *sf*, i.e. to the radius of its polar ball.

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, the centroid of the cluster. Performing a clustering consist in minimizing on the clusters Ω_i and on the generators ω_i an energy on the entire space :

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

where $\rho : \mathbb{R}^3 \to \mathbb{R}$ is a density function. Specifically, a clustering minimizes intra-cluster distances and maximize inter clusters distance.

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

- ≻ Given a set of generators $\{\omega_i\}_{i=1..k}$, optimize the definition of the clusters by minimizing the energy E.

- ≻ Given a set of clusters $\{\Omega_i\}_{i=1..k}$, 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).

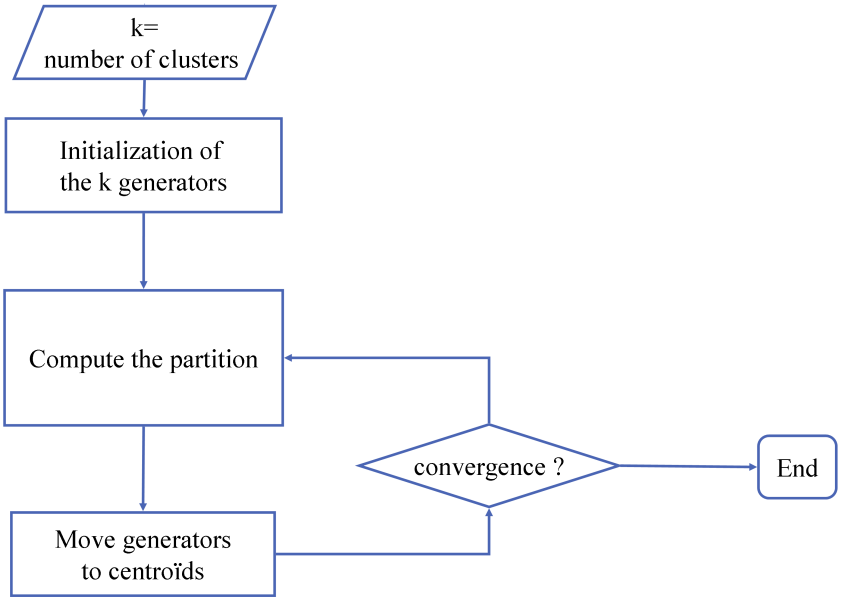


Figure 6.8: Clustering algorithm. Data : a set of n sample points and an integer k, the desired number of generators/clusters.

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

Density Function The density function, μ, is induced by a sizing function, σ. The sizing function defines a desired distance between a generator and its nearest neighbor (in the set of generator). Specifically, owing to the energy equidistribution property [DFG99], we know that the density function $\mu(x)$ must be proportional to $\frac{1}{\sigma(x)^{d+2}}$ to obtain a cluster density matching the field $\sigma(x)$ in a underlying space of dimension d.

In our work, we want a sizing function $\sigma(x)$ at a given point x similar to the sizing function $sf(\mathrm{x})$ defined in Section 5.1. As the medial axis, $M(S)$, is 2-dimensional for a surface S embedded in $\mathbb{R}^3$, we can define the density function μ with the following equation:

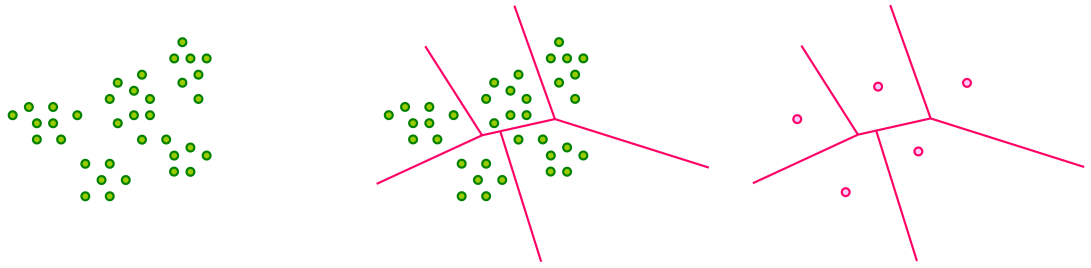
$$\rho(x) = \frac{1}{sf(x)^4}. \tag{6.7}$$

Note that for dx, the quadrature term represents the area of the elementary part of the medial axis surface associated to x.

6.1.2.2 The discrete case

Recall 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 performs 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 groups together all the points which hold a *same feature*, i.e. all the points close with respect to a given similarity measure (see Figure 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 (Figure 6.9(c)). This generator can be the centroid of the cell, the barycenter of the point set contained in the cell or one of the points of PC.

In the discrete case, the integral (6.5) is approximated by a discrete sum (6.8):

$$E = \sum_{\mathcal{C}_i} \sum_{v_j \in \mathcal{C}_i} \mu_j (v_j - c_i)^2 d(v_j) \tag{6.8}$$

where μ_j is a density value associated to each point v_j and $d(v_j)$ is the quadrature term defined as the area of the elementary part of the medial axis surface associated to v_j.

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

- ≻ Given a set of generators $\{c_i\}_{i=1..k}$, compute the clusters so as to minimize E,
- ≻ Given a set of clusters $\{\mathcal{C}_i\}_{i=1..k}$, compute the new set of generators which minimize E, i.e. the generators c_i are computed using:

$$c_i = \frac{\sum_{v_j \in \mathcal{C}_i} v_j \mu_j d(v_j)}{\sum_{v_j \in \mathcal{C}_i} \mu_j d(v_j)}. \tag{6.9}$$

Local Density Function The density value μ_j associated to the point $v_j \in PC$ is defined like the μ function in the continuous case (6.7). As the discrete value of the $sf(v_j)$ is r_j, the radius of the polar ball centered at v_j, we obtain the following equation:

$$\rho_j = \frac{1}{r_j^{d+2}} \tag{6.10}$$

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

$$\rho_j = \frac{1}{r_j^4} \tag{6.11}$$

Quadrature Term The quadrature term expresses the poles distribution. Specifically, 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 quadrature is performed in order to associate at each pole v_i its contribution. We want to compute the area of the elementary part of the medial axis which correspond to v_i see Figure 6.10(Left).

Let v_i be a pole. We compute the area of the intersection between the filtered medial axis and the Voronoi cell V_{v_i} of v_i in the Voronoi diagram of the poles. Let Vol_i be the volume of V_{v_i} and $d(v_i)$ the area of the elementary part of $M(S)$ corresponding to v_i. For dense sampling, most Voronoi cell are long and skinny like pencils (see Figure 6.10). Therefore, Vol_i can be approximated by $d(v_i)$ multiplied by the pencil length. Moreover, the pencil length can be approximated by the polar ball radius value r_i. This approximation allows us to approximate the area $d(v_i)$ with $\frac{Vol_i}{r(v_i)}$.

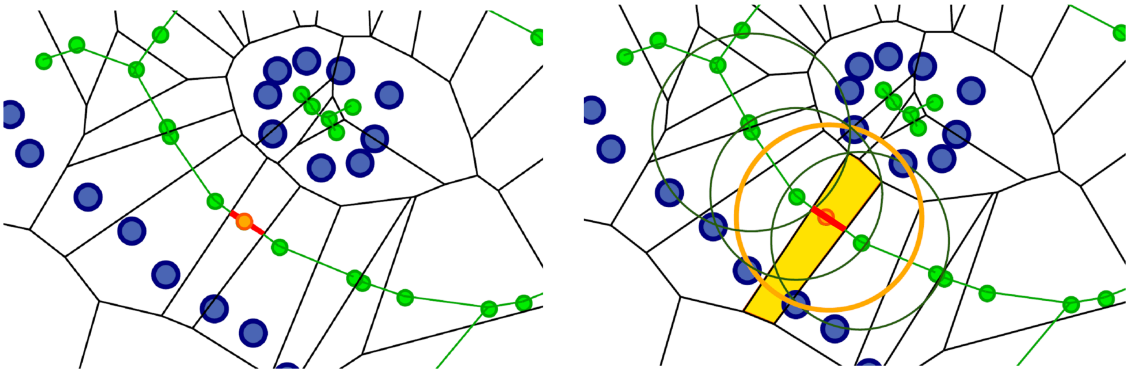


Figure 6.10: Quadrature term approximation (detail of the butterfly wing). Left: Voronoi diagram of the poles (green), we compute the quadrature term (red segment size) for the red point. Right: The volume of the orange Voronoi cell can be approximated by the product of the length of the red segment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

6.1.2.3 Implementation

Distance between two poles Two poles may be close but their respective radius can be very different. We want that poles with very different radii do not lie in the same cluster. To alleviate this issue, we compute the distance between two poles in 4D. Specifically, by adding the radius as a fourth coordinates for each pole:

$$d((v_i, r_i), (v_j, r_j))^2 = \|v_i - v_j\|^2 - (r_i - r_j)^2. \tag{6.12}$$

Initialization Problem The clustering algorithm is strongly dependent on the choice of the k initial generators. To overcome this problem, the initialization must take into account the desired density. We are going to perform several clustering successively such that, at each iteration:

- ≻ the number k of generator increases,
- ≻ 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. These $\frac{k}{2}$ points are distributed in the area with the highest 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 with the largest ratio. The algorithm stops when the number of clusters is equal to m.

Non Uniform Clustering vs Uniform Clustering The *uniform clustering* is performed using a uniform density function, i.e. $\rho(x) = 1\ \forall x \in \Omega$. We compare in Figure 6.12 the resulting set of centers after a uniform clustering (left) and a nonuniform clustering (right). The model used is a ball with two bumps (Figure 6.11). The two bump are subsampled, while the ball is oversampled. When a uniform clustering is performed, the center

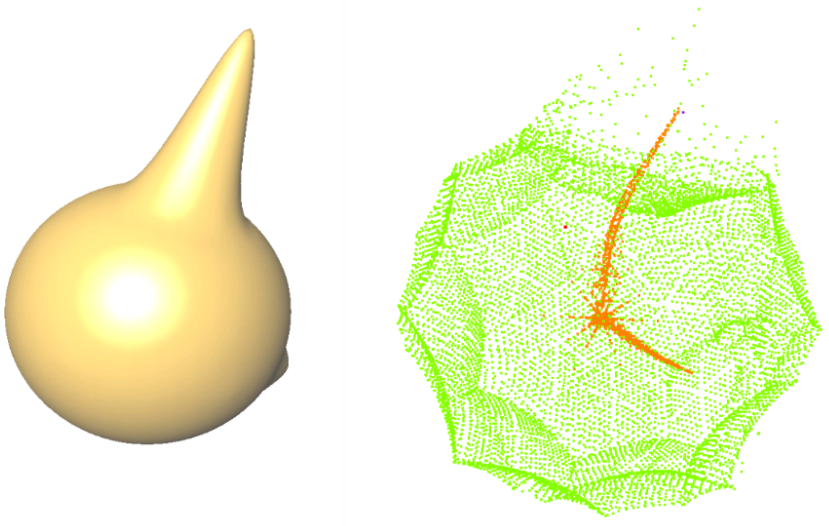


Figure 6.11: Non uniform clustering Vs uniform clustering. Left: a ball with two bumps. Right: all extracted poles (orange for inside poles and green for outside poles).

distribution remains related to the sampling density of the data points (Figure 6.12 right). In contrast, a nonuniform clustering provides us with a better distribution of the centers: the part of the medial axis where the local sizing field is small is densely sampled, see Figure 6.12(left). Figure 6.13 depicts the inside polar balls.

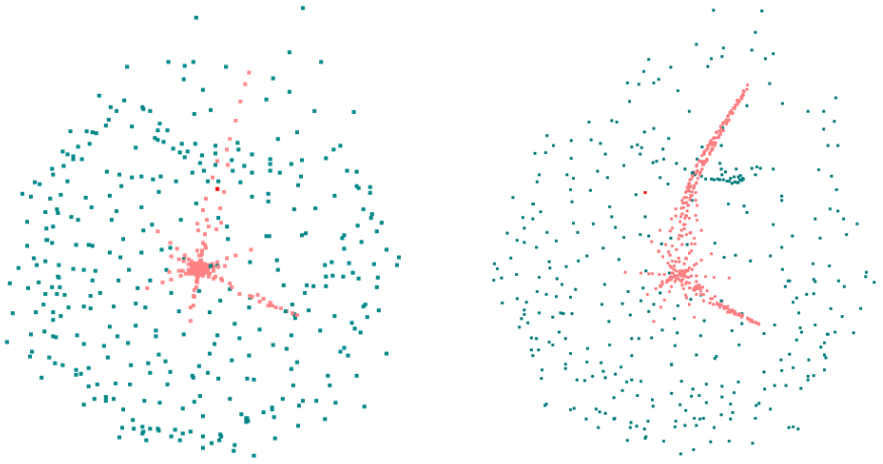


Figure 6.12: Location of the centers after a uniform clustering or our non uniform clustering. Left: the centers after uniform clustering. Right: the centers after nonuniform clustering. (red for inside centers and green for outside centers).

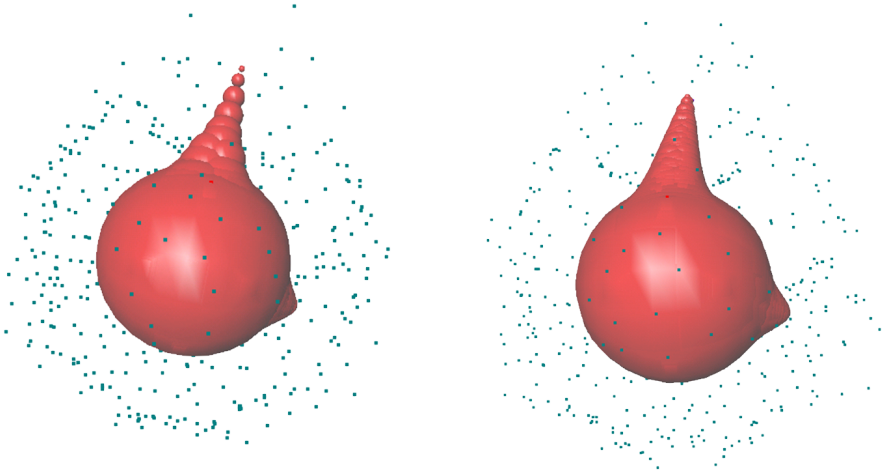


Figure 6.13: The maximal ball after a uniform clustering or our clustering. Left: centers after uniform clustering. Right: centers after nonuniform clustering (red for the inside polar ball and green for outside centers).

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.

In the pole filtering step the value for λ is fixed experimentally while the right value of λ is not really intuitive. Thus we propose another strategy simpler and more intuitive than the filtering/clustering. This strategy performs a greedy selection based on the overlapping rate of the polar balls.

6.2 Greedy Selection

Recall that m is the desired number of centers and PC is 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 density of the center set must be independent from the distribution of PC. The choice of centers must be adapted to the level of detail indirectly related to the allocated budget of centers m, see (Figure 6.1(Right).

As in [ACA07], the main idea is to perform a greedy selection of centers according to the sizing function, *sf*. In other words, the distance between a center c and its nearest neighbor in the center set must be proportional to $sf(\mathrm{c})$. 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 function. However, we know a discrete approximation of the medial axis as the pole set, PC. For each pole v, the sizing function is approximated by the radius of the polar ball centered at v, i.e. $sf(\mathrm{v}) = r_v$.

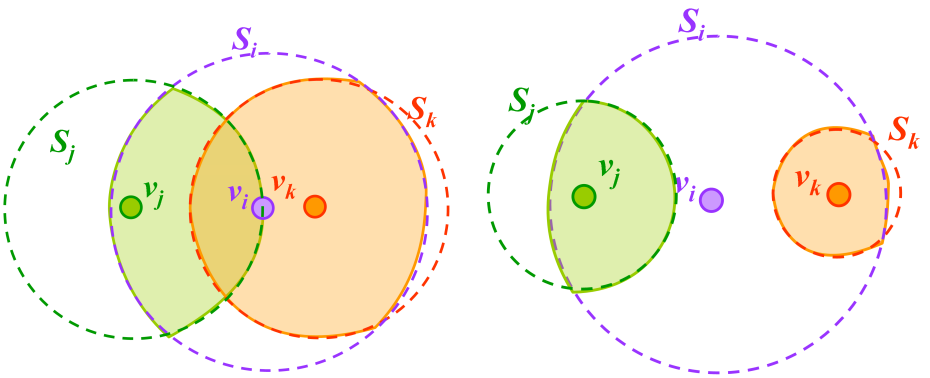
Our idea is to perform a greedy algorithm such that relevant poles are selected and redundant ones are disqualified. The density of the center points must be inversely proportional to the sizing function, i.e, the bigger the radii, the fewer poles (see Figure 6.2). In order to evenly distribute the centers, several poles are disqualified when a pole is selected as a center.

The poles are progressively added in the set of centers until the number of centers reaches m, iterating over the poles by decreasing polar ball radius. Let v_i be a selected pole. The closest pole to v_i may be disqualified, that is all poles, v_j, contained in the v_i-polar ball are candidate for the disqualification. If v_j is an unselected pole with a polar ball S_j which deeply intercepts v_i-polar ball S_i, v_j is disqualified. In order to test the deep intersection of S_i and S_j, we compare the volume of the intersection of the two polar balls $Vol(S_i \cap S_j)$ with the volume of the smallest polar ball $Vol(S_j)$ (Figure 6.15).

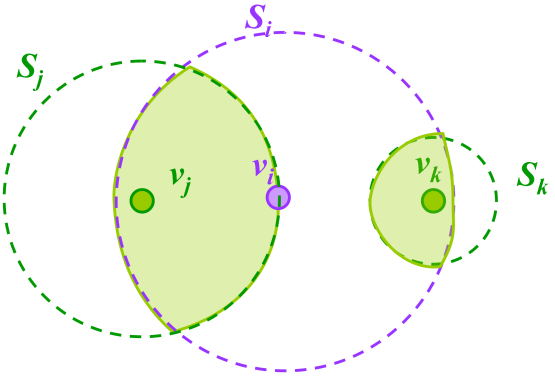
Note that, as illustrated by Figure 6.14, either the distance from the pole v_j and v_i (Figure 6.14(a)) or the difference between the radii of the corresponding polar balls (Figure 6.14(b)) are not discriminant.

A user defined overlapping rate threshold ρ allows us to decide if a pole is disqualified or not. When the number of selected centers reaches m, several non-disqualified poles may still not by selected. These poles are associated to a polar ball with a too small radius with respect to the user-defined number of centers. This way we obtain a center set adapted both to the desired density and to the level of detail given by the center budget (Figure 6.14(c)).

Algorithm First, we sort the poles according to their radii. We then select the poles with the larger radii and disqualify the poles which polar ball deeply intersect the polar ball of the selected pole. Specifically, let v be the selected pole and S be the polar ball of S_v. For each pole v_j in S_v, if the volume of the intersection of S_{v_j} and S_v, the polar ball of v_j, is greater than $\rho \times Vol(S_{v_j})$ the pole v_j is disqualified; where ρ is a threshold for the



(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 allocated budget of centers is sufficient.

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

overlapping rate. The set of disqualified poles $D_p(v)$ is defined as:

$$D_p(v) = \{v_j \in S_v | Vol(S_{v_j} \cap S_v) > \rho \times Vol(S_{v_j})\} \tag{6.13}$$

As shown Figure 6.16, the value of ρ determines the level of detail. A trade off between the level of detail and the overlapping rate of the polar balls may be found. Note that when $\rho = 1$ (Figure 6.16(a)) the m selected centers are the m poles associated to the largest polar balls.

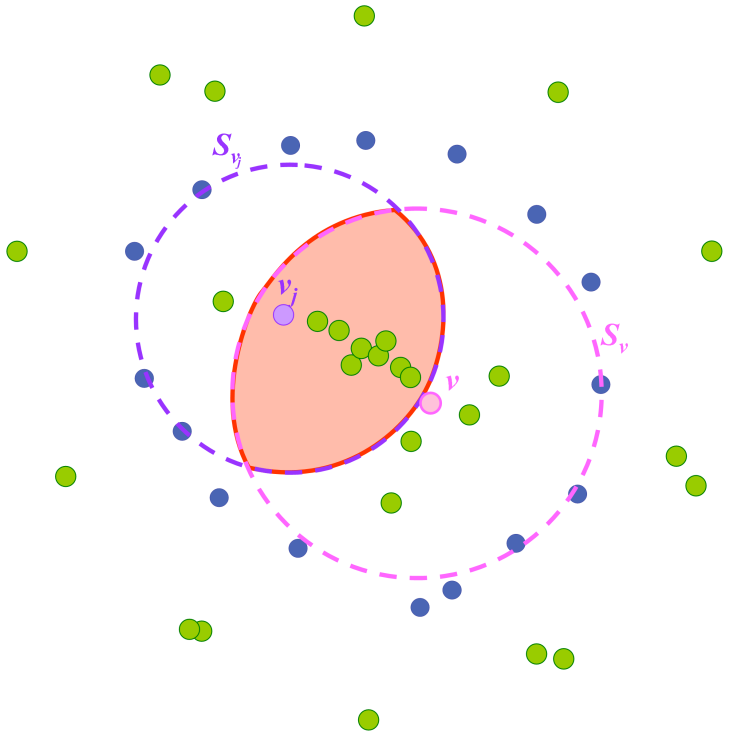
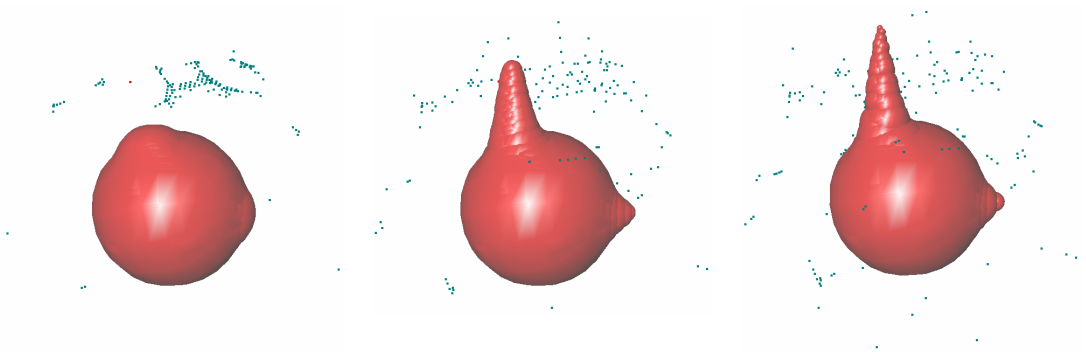


Figure 6.15: Disqualification criterion for greedy selection. Given a selected pole v and a set of poles v_j, the volume of the red area is compared to the volume of S_{v_j},the smallest polar ball among $S - v$ and S_{v-j}.



(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 polar balls (red) and outside centers (green).

Chapter 7

Algorithm Implementation and Optimization

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 ourselves to the case where the surface divides the space into two subspaces : 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 functions ϕ centered at a set of center points c_j:

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

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

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

given a set of N constraints $\{x_i\}_{i=1..N}$ where the function f is known. The minimization consist of 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 center points from the set of poles;

5. Assemble the matrix G;

6. Solve the linear system.

We provide details about the implementation of the step 3, the pole classification algorithm, and 5, the matrix assembly step.

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]. For the visualisation, we use a Delaunay-based surface mesher [RY06].

7.1 Poles Classification

Recall that the set of *poles* is a subset of Voronoi vertices. Given a point $p \in P$, let $V_p = V_{P,p}$ be its Voronoi cell in the Voronoi diagram of P. Two poles are extracted for each bounded cell V_p (see Section 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 the further away from p in the opposite half space of v_1. Note that if the Voronoi cell V_p is unbounded p owns zero pole or a unique pole, the Voronoi vertex the further away from p in V_p.

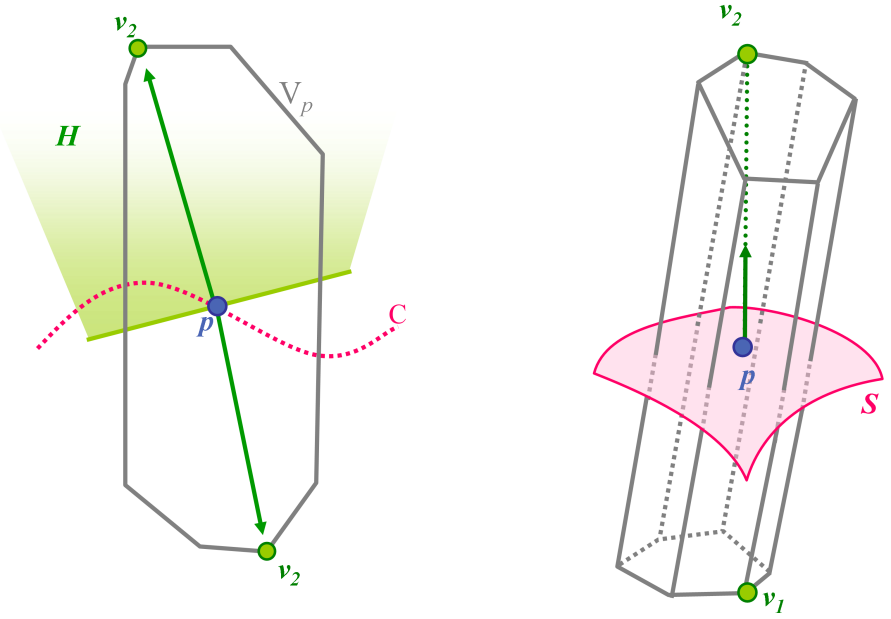


Figure 7.1: Voronoi cell in 2D and 3D

If the sampling is dense enough, the vector $p\vec{v}_i$ is a good approximation of the normal to the surface at p (Figure 7.1). Thus, if the sampling is dense enough v_1 and v_2 lie on each side of the surface.

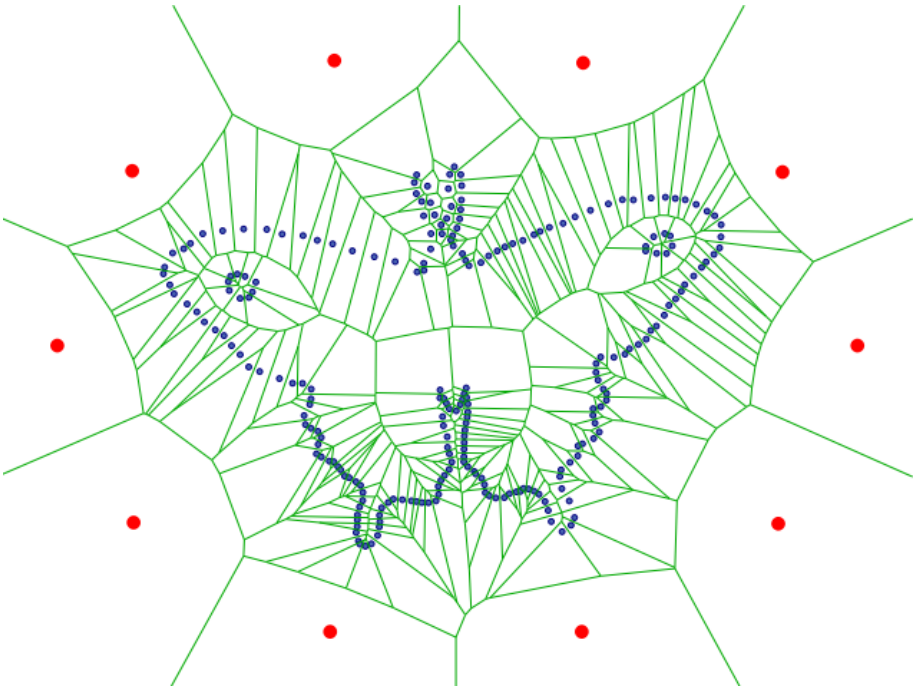


Figure 7.2: Voronoi diagram of input points set augmented by an inflated instance of the convex hull points (red)

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

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

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 first case, the probability of having the poles lie on the two opposite sides of the surface is high. Conversely, in the second case the poles v_i and v_j can lie either on the same side or not but it is more probable that the poles lie on the same side (Figure 7.3).

The main idea of the labeling algorithm is to associate to each pole temporary labels, in and out, and a probability measure on the temporary labels. Specifically, the labels and their probability are propagated through the pole graph. The points of B are used to initialize the algorithm.

In [ACK01] (see Section 2.1.2), the labels are propagated through the pole graph. The

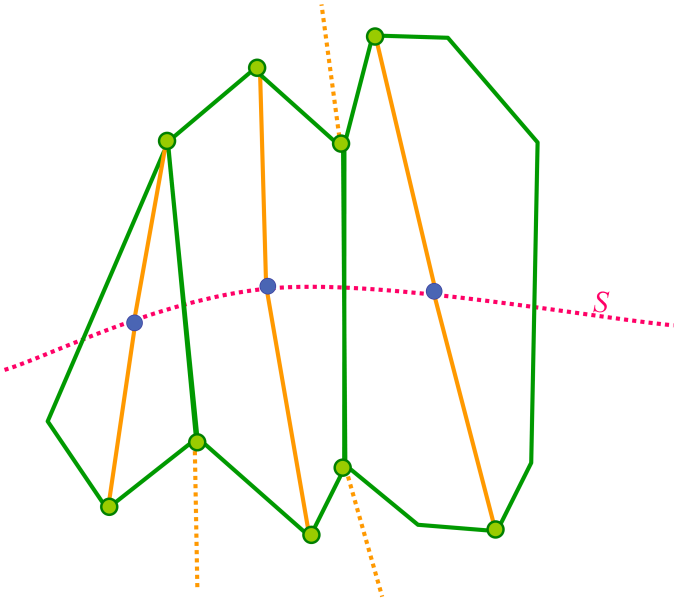


Figure 7.3: Edges of the poles graph in 2D. There are two kinds of edges joining two poles. Orange edges when two poles share a vertex (case 1) and green edges (Voronoi edges) when the poles are neighbors in the Voronoi diagram (case 2).

algorithm uses a priority queue containing the poles with a temporary label. The priority is based on the label probabilities.

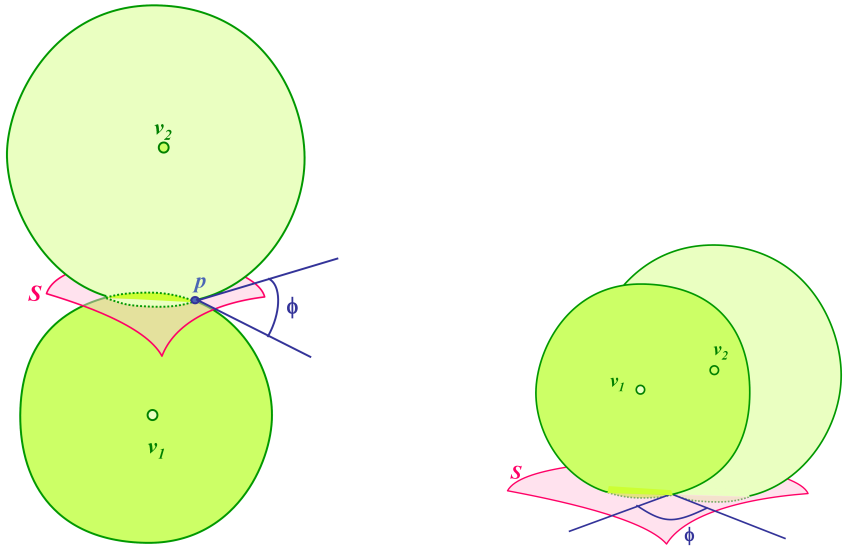
In order to initialize the algorithm, the poles of the bounding points are labeled as outer with a probability one. Then the pairs (label, probability) are propagated until all poles are labeled. The pole v with the highest probability is popped out of the queue, and gets a final label. Then, the labels of the neighbors of v in the poles graph are updated and the priority queue is updated (see Algorithm *PolesClassification*).

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

- 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 (Figure 7.4(a)).

- Case 2: the temporary label of v_2 is fixed to the v_1 label with a probability proportional to the cosines of ϕ (Figure 7.4(b)).

This classification method assumes some conditions over the sampling: ϵ-sampling (see Appendix B.2). Recall a sampling 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.



(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.

Algorithm *PolesClassification*
Input: The Voronoi Diagram of the input points P
Output: A set of labeled poles

for all poles v
  do initialize $in(v) = out(v) = 0$
    insert p in the priority queue $\mathcal{Q}$
for each pole v adjacent to points of B
  do out(v) = 1
    Update Priority(v) in $\mathcal{Q}$
while Q is not empty
  do Remove the top element v of $\mathcal{Q}$
    if $in(v) > out(v)$
      then $label(v) = in$ and $tmp(v) = in(v)$
      else $label(v) = out$ and $tmp(v) = out(v)$
    for each input point p of which v is the pole
      do let u be the other pole of p
        $opp(label(v))(u) = max(tmp(v) * cos(\phi); opp(label(v))(u))$
        Update Priority(u)
    for each deeply intersecting neighboring poles u
      do $label(v)(u) = max(tmp(v) * cos(\phi); label(v)(u))$
        Update Priority(u) in $\mathcal{Q}$

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 have been introduced to discards non confident poles. If the cells are not long and skinny, they are rejected.

In the Eigen Crust [KSO04] (see Section 2.1.6), the classification is performed by computing 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 subgraphs containing inside and outside poles.

In our algorithm, we perform a variant, due to F.Cazals (internal communication), more efficient and robust against noise. In the two approaches presented above, only the poles are taken 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 a node is a pole with a temporary label and a probability or a point with an temporary oriented normal and a probability.

7.2 Matrix Assembly

Given a set of N constraints, $\{x_i\}_{i=1...N}$ associated to 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 of solving linear system $G\alpha = F$, where G is given by (7.6), in the least squares sens, i.e., by solving the system:

$$G^t G \cdot \alpha = G^t F. \tag{7.5}$$

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 equivalently $G^t G$.

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

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

where the symbol $+$ means $(x)_+ = x$ if $x > 0$ and $(x)_+ = 0$ otherwise. S_j is the supporting ball of ϕ_j, centered at 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 at c_j

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

7.2.1 Trivial Method

A trivial method would constructs Matrix G (7.6) by computing $\phi_j(x_i)$ for all centers c_j and all constraints x_i. Then, we can compute Matrix $A = G^t G = [a_{i,j}]$, $i, j = 1..m$ as

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

7.2.2 Selective Method

As the radial basis function ϕ_j are compactly supported, the matrix $G^t G$ is sparse although less than G.
About ten percent of matrix coefficients being non-zero, it is worth computing only the non zero coefficient instead of performing an exhaustive iteration. We can note that $a_{i,j}$ (7.9) is zero when the supports S_i and S_j do not intersect (Figure 7.5).

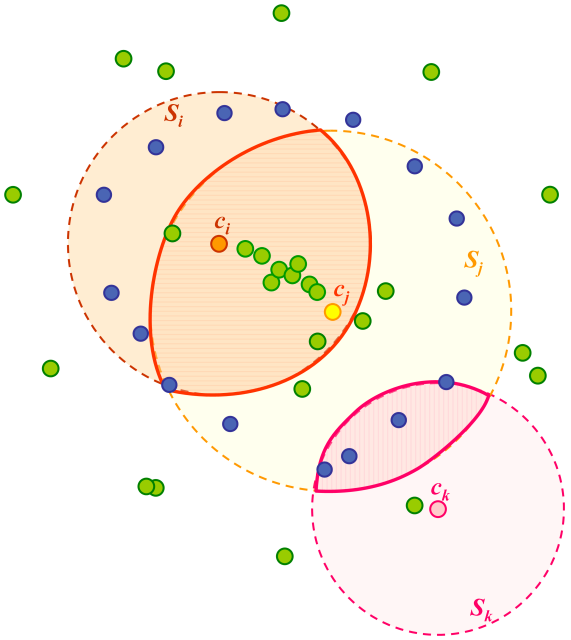


Figure 7.5: Input points (blue) and centers (green). The support of ϕ_i and ϕ_j (S_i and S_j) intersect and share several centers: $a_{i,j} \neq 0$. The support of ϕ_i and ϕ_k (S_i and S_k) does not intersect: $a_{i,k} = 0$.

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

Our idea is to iterate over the centers and to 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 retrieve the constraints in S_j we construct a *kdtree* over the constraints. This kd-tree allows us to avoid visiting all constraints. The constraints are visited according to their increasing distance to c_i and the algorithm stops when the current constraint x_i is such that $\|x_i - c_j\| > \|S_j\|$.

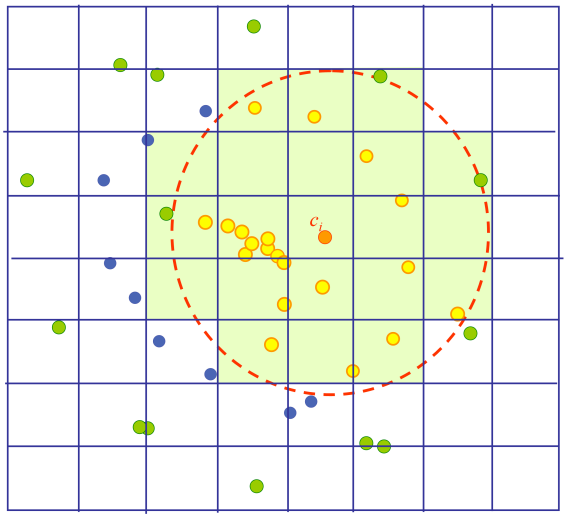


Figure 7.6: Given a center c_i, a list of constraints (yellow points) is constructed using kd-tree. Only the constraints in the green area are visited.

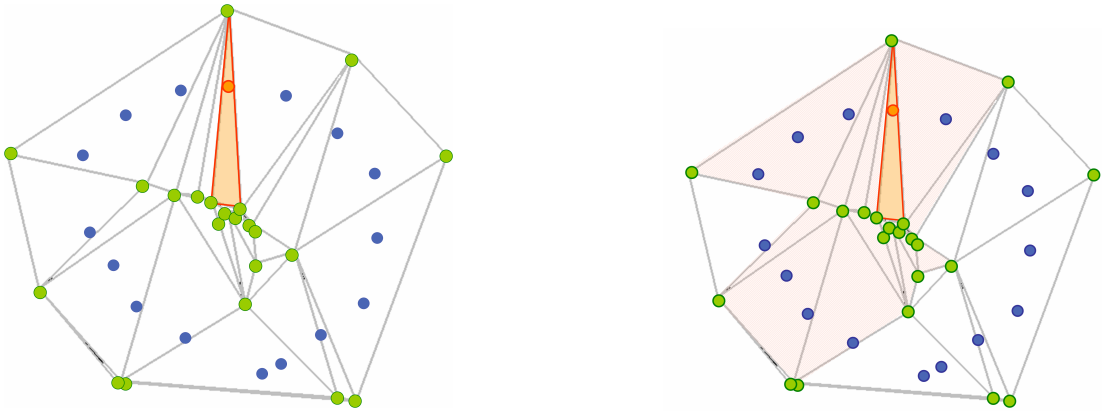
Note that the values $\phi_i(x)$ are precomputed for all $x \in S_i$ during the computation of the diagonal term. The centers are sorted according to their support and the centers with the smallest support are processed first.

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 center c_i with a large support S_i which contains n constraints (with n sufficient large). Suppose that S_i slightly intersects S_j, the support of c_j, with 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, which lead to n tests instead of s points in the sum (7.9).

Therefore, it may be more efficient to use the dual method. The idea now is to go through the constraints and to search for the centers which contain x in their support. In the following, we call these centers *x-relevant centers*.



(a) the point is located in DTC.

(b) Propagation in the triangulation of centers.

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

The *dual* method consists in iterating over the constraints, and 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 (Figure 7.7(a)).

To avoid a greedy algorithm which tests all centers for each constraint, 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 basis for the traversal algorithm in order to collect all centers which contain the current constraint in their support (Figure 7.7(b)).

A stack of candidate cells is constructed to collect the *x-relevant centers*. A candidate cell may satisfy these two following conditions:

- at least one on its vertices is a *x-relevant center*
- at least one on its vertices has not been visited yet for the current constraint x

The propagation in the triangulation of centers $\mathcal{D}_C$ stop when there are no more candidate cells. This strategy assumes that the *x-relevant* centers for a given constraint are located in a connected area.

Data Structures :

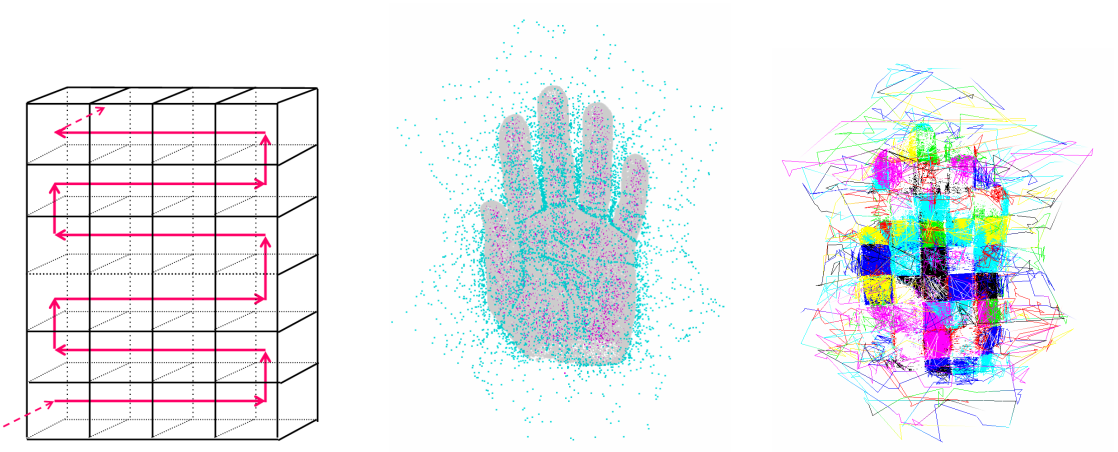
The Delaunay triangulation of the centers, $\mathcal{D}_C$, allows to structure the center set with neighborhood relationships. Given a constraint x, $\mathcal{D}_C$ allows for collecting the centers close to x by a traversal algorithm.

The localization in $\mathcal{D}_C$ may be expensive if the search starts 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 along a *space filling curve*. A space filing curve is a line passing through every point in a space, in spatial-coherent order (see Figure 7.8(a)).

Algorithm : The algorithm *MatrixConstruction2* iterates over the constraints $\{x_i\}$. For a given constraint x_i, we want to find all the centers which contain x_i in their support.

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.  Initialize A[i][j] and b[i] to zero, i, j = 1..m.
2.  for all sfc box sfc_i
3.      do for all constraints x ∈ sfc_i
4.          do Locate x in D_C.
5.             Let C the cell which contains x.
(* Seek after the x-relevant centers *)
6.             push C is a cellStack
7.             while cellStack non empty
8.                do pop the first cell C_k in cellStack
9.                   for all neighbor tetrahedra T of C_k
10.                     do for each vertex c_i of C_||
11.                           do if c_i not visited yet for x
12.                                 then Tag c_i as visited
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 φ_i(x)
17.                        if at least one of vertices is x-relevant
18.                           then add T in cellStack
19.             for all center pairs (c_i, c_j) in x-relevant centers list
20.                do Update the values A[i][j]
21.
```



(a) Space filling curve in 3D.

(b) Set of constraints points.

(c) Space filling curve on the constraints.

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

Chapter 8

Results

This chapter presents a set of experimental results produced by our reconstruction algorithm for models of various complexity, topology and sampling density. We first provide typical timings for all steps of our algorithm, and assess its main features as expected from a RBF-based reconstruction technique. We then show the relevance of choosing Voronoi vertices at centers, and show results with varying parameters. The models used in our experiments are either synthetic data, such as the *Knot* model, or input point sets acquired using a laser scanner, such as the *Bimba* or the *Hand* model.

8.1 Algorithm Sequence

Figure 8.1 depicts all steps of our algorithm on a 2D point set:

1. Poles are extracted and classified from the Voronoi diagram (Figure 8.1(b));
2. Poles are filtered (Figure 8.1(c));
3. Poles are clustered into centers (Figure 8.1(d)) in order to select a user defined number of centers. The set of centers is relevant if the hull of the set of inner (resp. outer) polar balls is a good approximation of the shape (Figure 8.1(e));
4. The 2D implicit function is computed and its zero-level set is extracted (Figure 8.1(f)).

Note that superfluous connected components may appear on areas with no data points nor centers, as shown by Figure 8.1(f) (bottom right).

As a typical example for our algorithm, we detail the timings of each reconstruction step for the *David head* model with 100K input points. The original point set has been randomly subsampled, see Figure 8.2.

1. Point insertion to the Delaunay triangulation: 6.3s;
2. Extraction of 190K poles: 3.4s;
3. Classification (94K inside and 93K outside poles): 7s (Figure 8.2(b));
4. Greedy selection of 20K centers with $\rho = 0.2$: 8s (Figure 8.2(c));

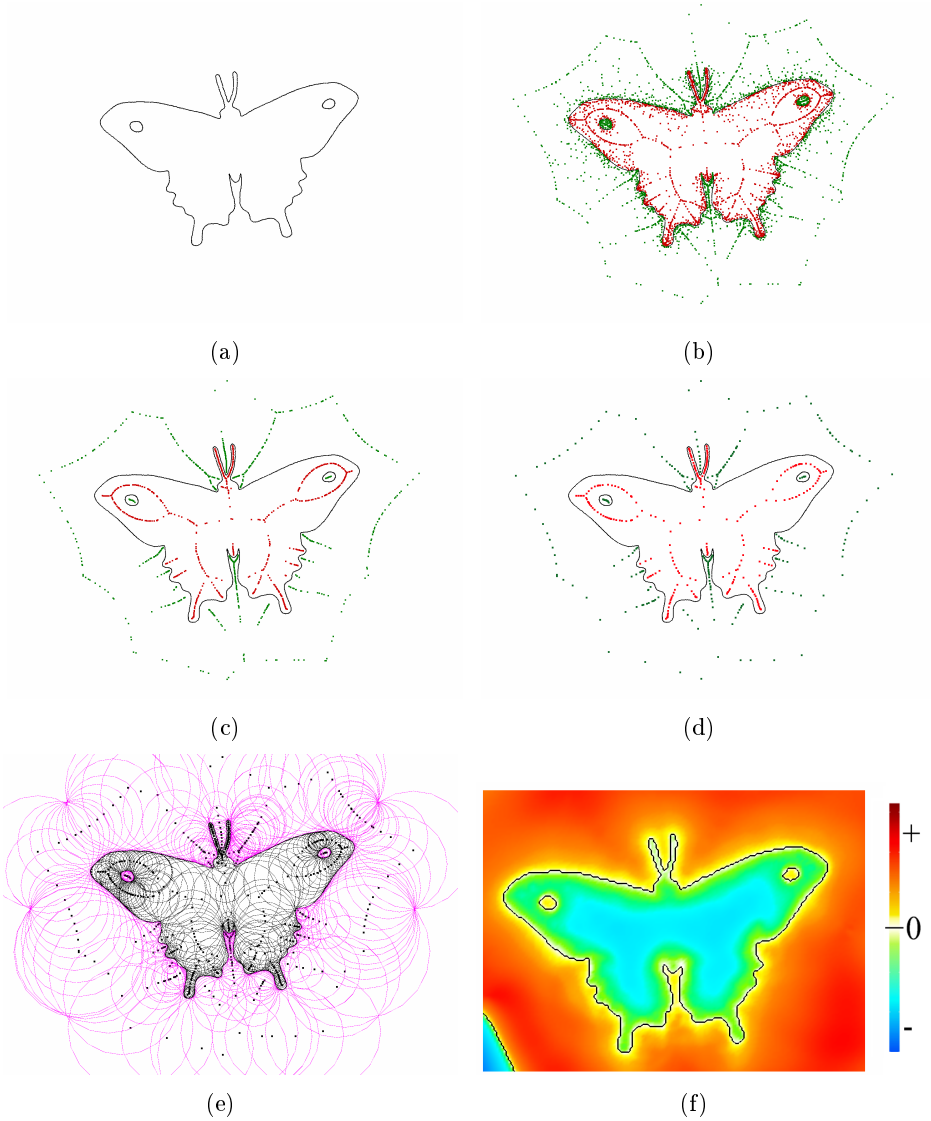


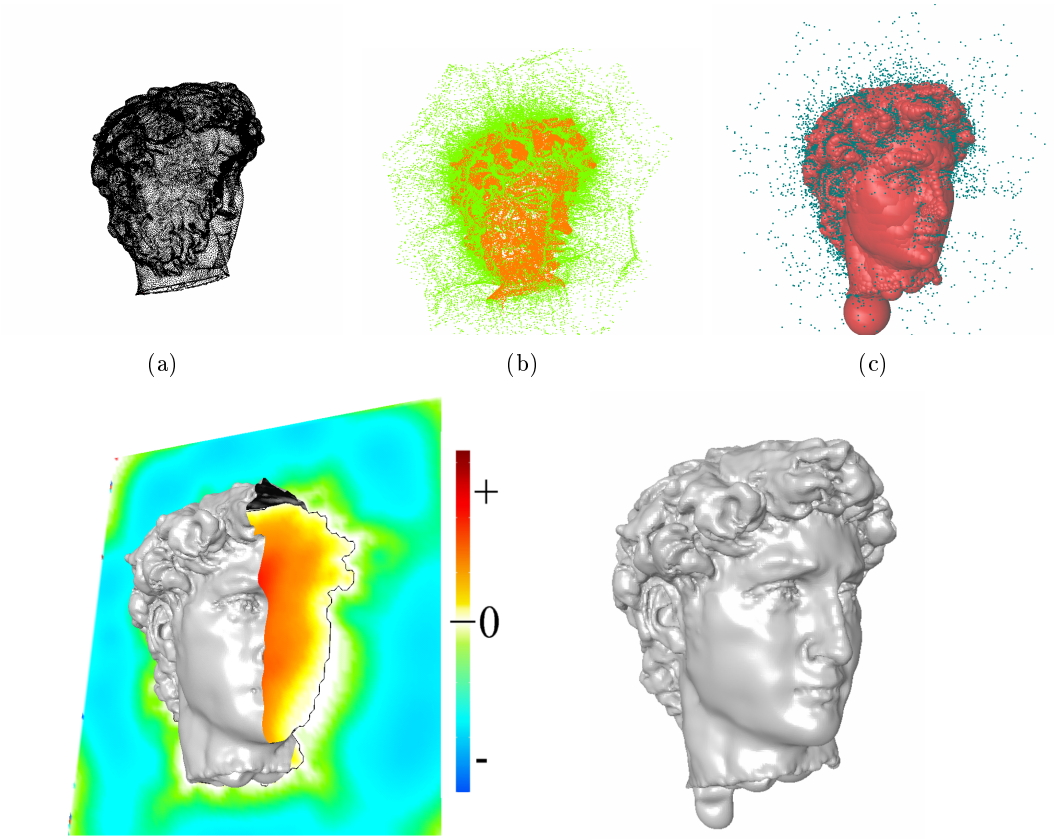
Figure 8.1: Algorithm sequence in 2D with filtering and clustering. (a) 100K input points. (b) 4K poles (green outside, red inside).
(c) 1500 filtered poles, filtering with $\lambda = 0.005$ and $\lambda_r = 0.5$.
(d) 400 selected centers (red inside, green outside).
(e) 400 selected centers with their polar balls.
(f) Reconstructed curve (black). The color map represents the function values (cold tones for positive, hot tones for negative and white for zero values).

5. Assembling the linear system: 680s (98% of zero coefficients in Matrix G^tG);

6. Solving the linear system: 78s.

The least square mean error is about 9.4×10^{-8} for 20k centers. Our final RBF-based representation is composed of a list of center coordinates c_i with their support size s_i and their coefficient α_i. In our current implementation, most of the time is spent assembling the linear system (86% of the total time). Specifically, most of the time is spent finding all pairs of centers whose supports intersect a constraint, even when using a 3D Delaunay

triangulation to avoid a naive exhaustive search (see Section 7.2).



(a) (b) (c)

(d) (e)

Figure 8.2: Algorithm sequence in 3D with greedy selection.
(a) 100K input points.
(b) Pole set (green outside, orange inside)
(c) Selected centers with their inside polar balls (red inside, green outside)
(d) Reconstructed surface with a cutting plane showing the implicit function.
(e) Reconstructed surface.

High Genus As expected from an implicit-based technique our algorithm reconstructs 3D models with non-trivial topology such as the genus-65 *Filigree* model, see Figure 8.3.

Hole Filling As our approach produces a watertight surface, it is well suited to fill holes as illustrated by Figure 8.4.

Filling small holes as shown by Figure 8.4 is relatively easy, and can be performed by Delaunay-based techniques as well [DG03]. More challenging examples occur for point sets with large holes due to occlusion during acquisition or due to non watertight object. In the example depicted by Figure 8.5, a large piece of the surface patches the back of the reconstructed *Julius mask* model. However, as shown by Figure 8.2(e), large holes can lead to misclassified poles, and hence to bumps in the reconstructed surface. This artifact is explained by the sampling condition not suited to correct pole classification, see Section 7.

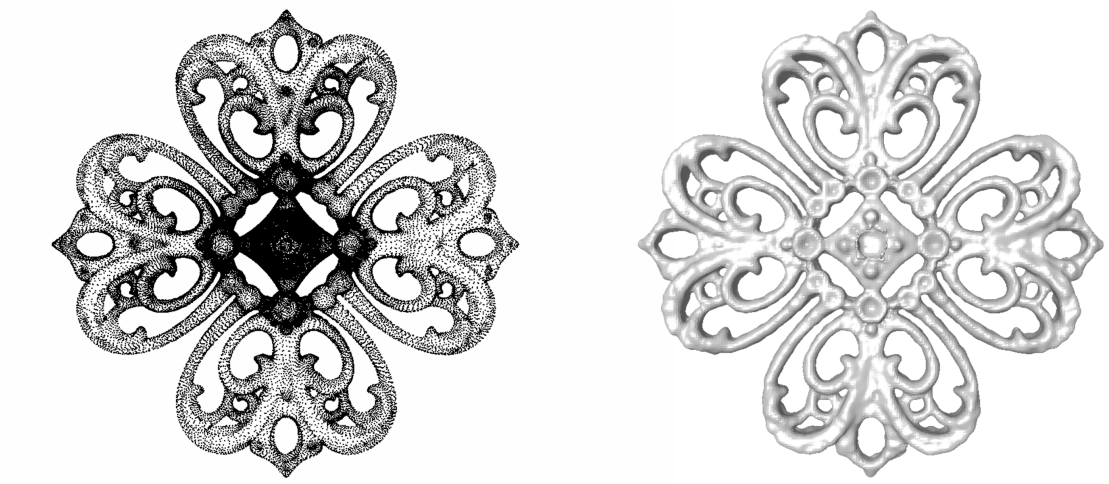
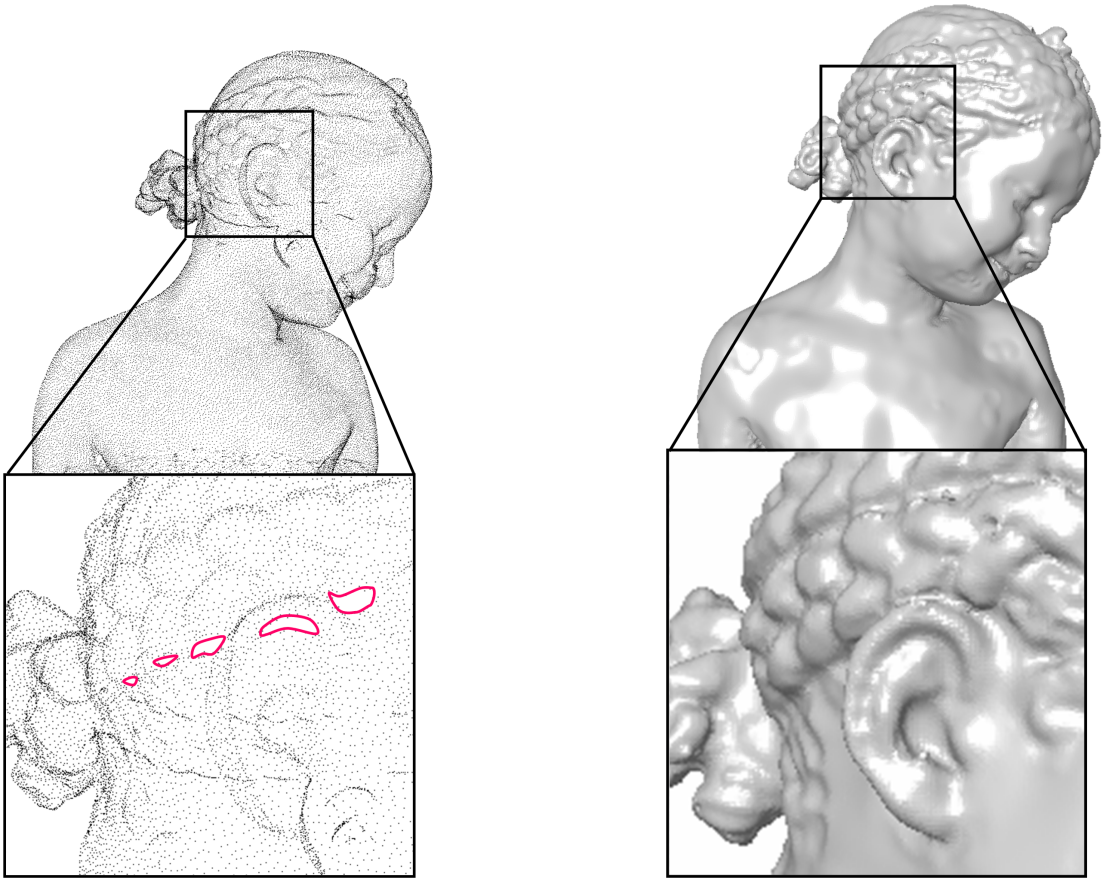


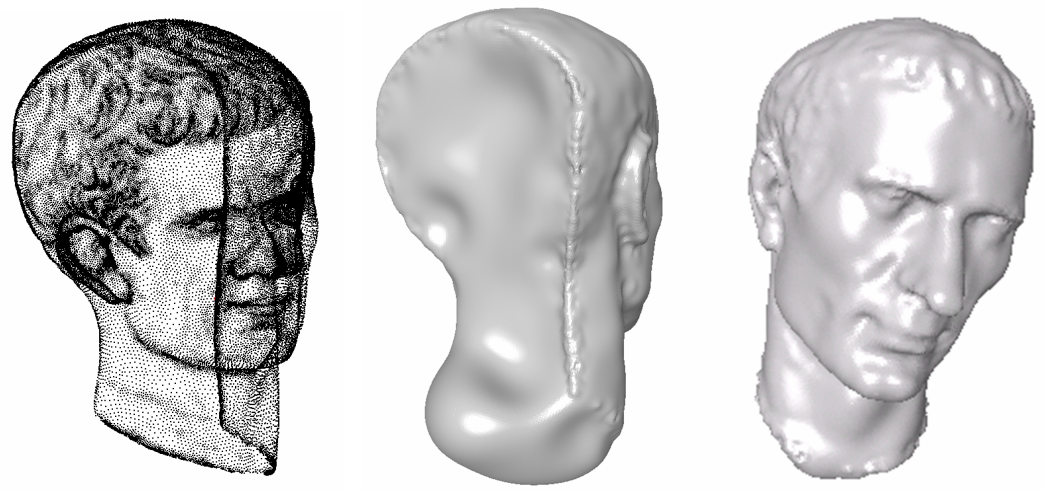
Figure 8.3: Reconstruction of the *Filigree* model (80K points) with 13K centers. Left: 80k input points; Right: reconstructed surface (fitting accuracy: 2.8×10^{-6}).



(a) Point set with holes (pink) coming from several hidden areas.

(b) Reconstructed surface with holes filled.

Figure 8.4: Hole filling. Reconstruction of the *Bimba* model (100K points) with 11K centers obtained by clustering.



(a) Point set sampled on a surface with boundary.

(b) Reconstructed watertight surface.

Figure 8.5: Reconstruction from a point set sampled on a surface with boundary. *Julius mask* model (43K input points) reconstructed with 18K centers obtained by greedy selection.

Smoothness As the radial basis function chosen is smooth, our technique is limited to the reconstruction of smooth surfaces. A point set sampled on a surface with sharp creases is reconstructed as a smooth function, see Figure 8.6.

Noise Figure 8.18(d) illustrates a challenging example with a substantial amount of registration noise (three range maps have been registered). Moreover, the sampling is highly non isotropic and non uniform due to the acquisition system.

8.2 Voronoi-Centered RBF

The relevance of choosing Voronoi vertices as centers is shown graphically by Figure 8.9. We plot the fitting accuracy against the number of centers both for our method and for the standard method where constraints and centers coincide. To evaluate the fitting accuracy, we use the *Taubin distance* [Tau94] from the input points to the computed function:

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

To the first approximation order this error sums the squared Euclidean distances between the input point set P and the zero level set of the computed function f. Since the gradient can vanish or go to infinity with compactly supported basis functions, we

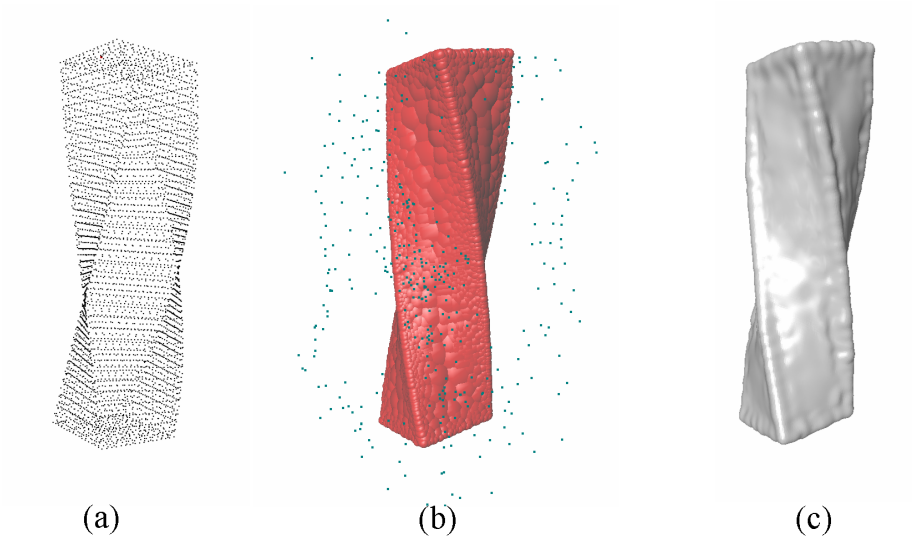
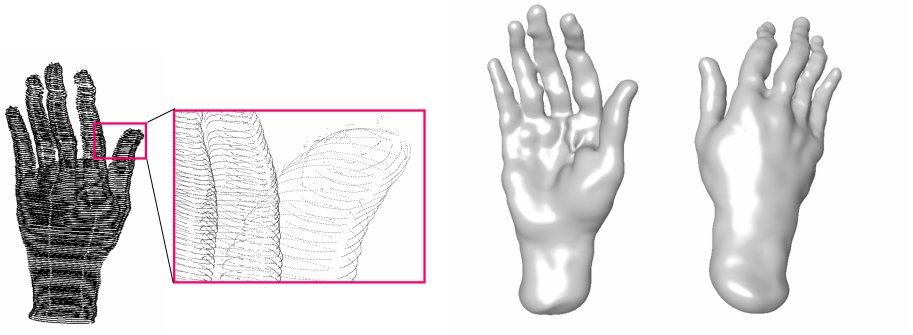


Figure 8.6: Piecewise smooth surface reconstruction. 14K input points sampled on a piecewise smooth surface. (a) Reconstructed with 2K centers. (b) 2K centers obtained by filtering ($\lambda = 0.1$) and clustering. (c) Reconstructed surface.



(a) Input point set. (b) reconstructed hand.

Figure 8.7: Noisy hand model. (a) 90K input points obtained by registering three range maps. (b) Reconstructed surface with 2K centers.

incorporate a function Γ (see Figure 8.8) 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 :

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

We compare our approach against the standard RBF method where centers coincide with input points. To ensure that we compare with identical number of centers, we randomly subsample the input points for the standard approach. The off-constraint points are taken along the normals estimated at the points.

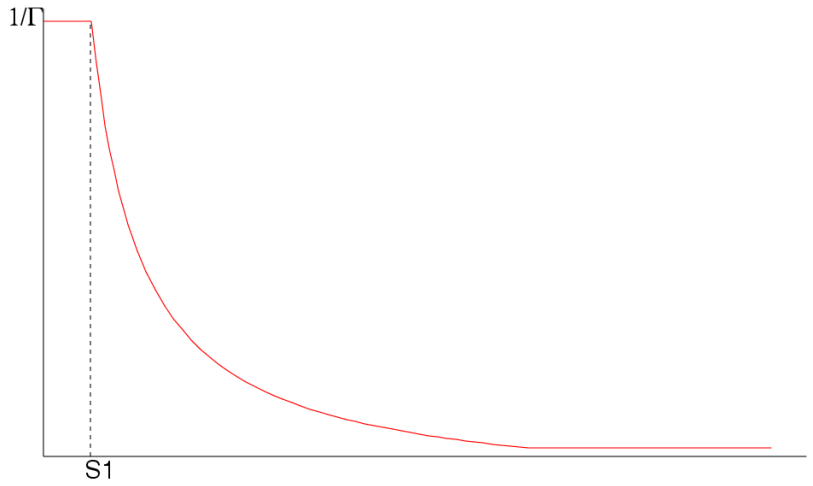


Figure 8.8: Error function.

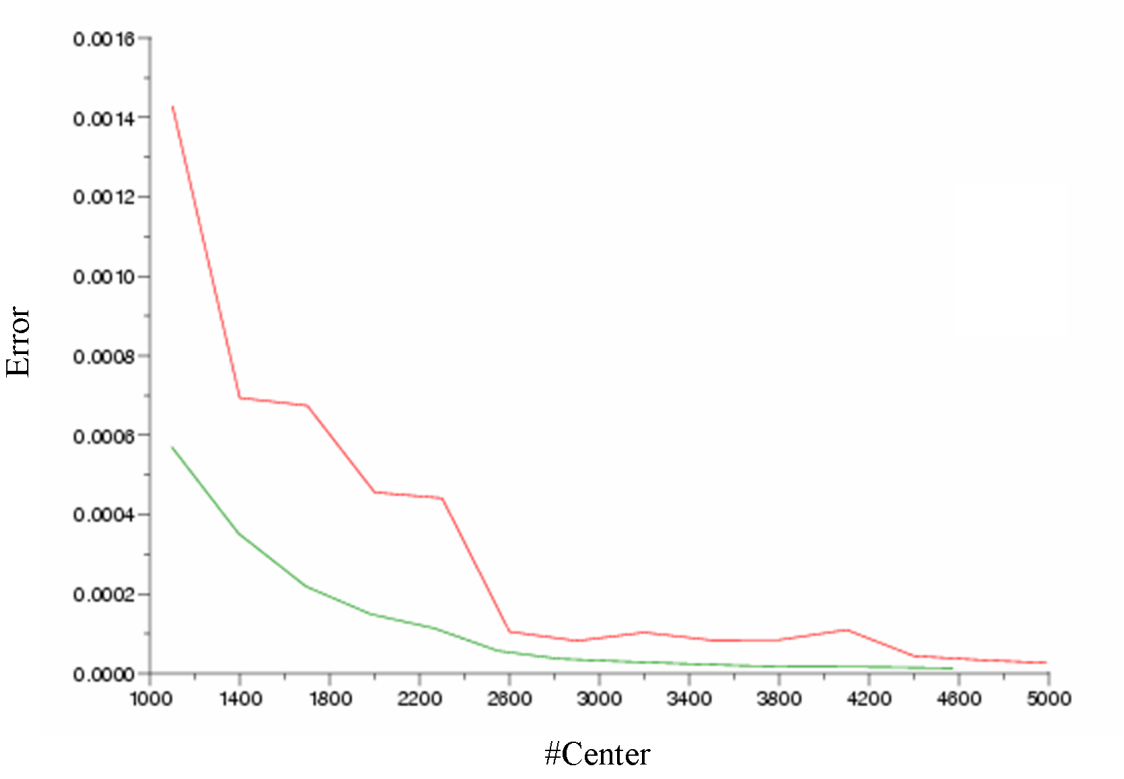


Figure 8.9: Fitting accuracy against number of centers (from 1K and 5K) for the Dinosaur model (15k input points). The red curve corresponds to the standard method. The green curve corresponds to our approach.

Figure 8.10 illustrates several reconstructions of the Dinosaur with increasing number of centers corresponding to the fitting accuracy plotted in Figure 8.9 (green curve).

As Figure 8.11 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

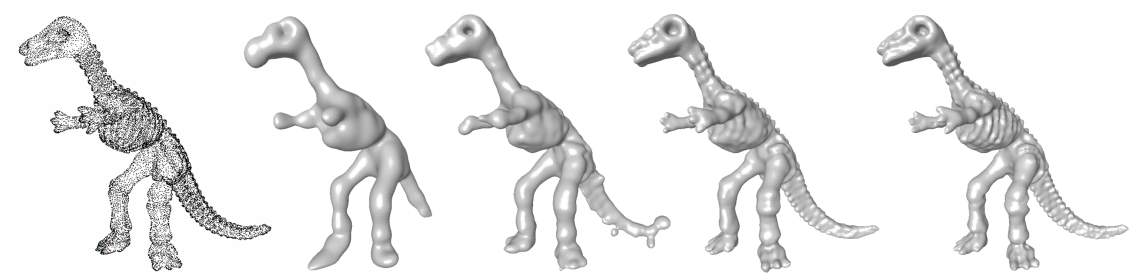


Figure 8.10: Reconstruction sequence of the Dinosaur with increasing number of centers. From left to right: input point set (15K), then reconstruction with 1K, 2K, 3K and 5K centers selected among the poles.

input data points, the function is only defined in a tubular neighborhood of the sampled surface. This property allows location queries in a large neighborhood around the point set. In addition, it allows for performing surface reconstruction even with large holes as shown by Figure 8.5(b)). In contrast, the *adaptive CSRBF* method [OBS04] creates a watertight surface by offseting the front side of the mask, see Figure 8.12.

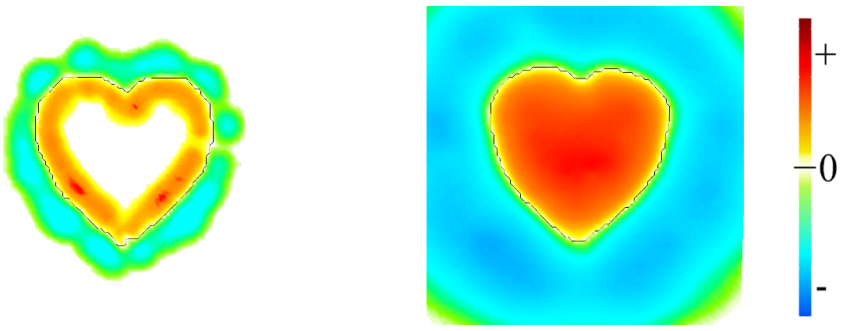


Figure 8.11: Reconstructed function. The colors represent the function values (cold tones for positive, hot tones for negative and white color for zero values). Left: reconstructed function for the standard approach; The function vanishes in a tubular neighborhood around the point set. Right: reconstructed function for our method.

The adaptive CSRBF produces a complex shape representation which includes the centers coordinates, the quadric coefficients, and the coefficients as well as the support size of the basis functions. In contrast, our technique only requires storing the centers and coefficients.

8.3 Filtering and Clustering

The pole filtering step is used to adapt the level of detail to the user-defined number of centers (see Figure 8.13), as well as to improve robustness against noise. It also shows the impact of filtering when the allocated budget of centers is low.

For the hand model, $10k$ centers are not sufficient to reconstruct all the details. Cluster-

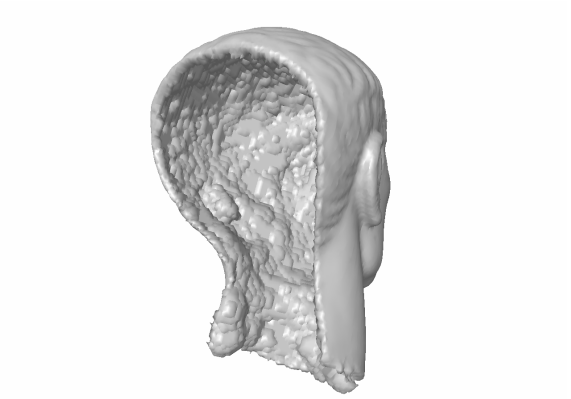
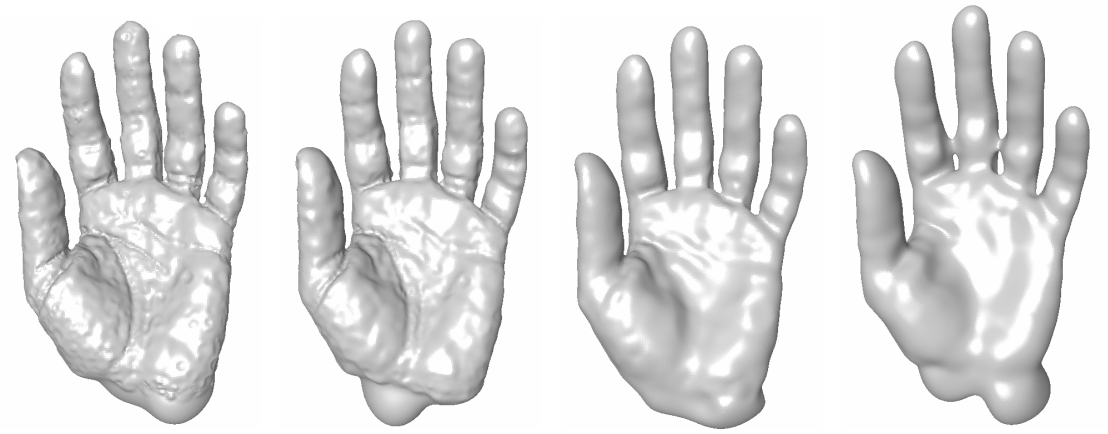


Figure 8.12: Reconstruction with adaptive CSRBF from a point set sampled on a surface with boundary. *Julius mask* model (43K input points) reconstructed with 18K centers.



(a) $\lambda = 0.01$ (b) $\lambda = 0.01$ (c) $\lambda = 0.02$ (d) $\lambda = 0.03$

Figure 8.13: Impact of the filtering step over the Hand model (50K input points). The desired number of centers ($m = 10K$) is obtained by clustering. To get a better insight into these parameters, the diagonal length of the bounding box of the input point set is 1.2 and the maximal radius value is 0.4).

ing promotes the centers associated to the small polar balls, as the specified center density is proportional to the sizing function $sf(\mathrm{x})$. As a result there are more centers in the areas where $sf(\mathrm{x})$ is small than in the areas where $sf(\mathrm{x})$ is large. Figure 8.14 illustrates the center radii distribution for various filtering parameters followed by clustering.

The peak at the origin is due to the clustering which promotes small radii. The filtering step allows smoothing down the peak and thus obtaining a distribution adapted to the allocated budget of centers. The curves plotted in Figure 8.15 show the radii distribution for all reconstructions illustrated by Figure 8.17, and the radii distribution for all poles with, and without filtering.

To assess the impact of filtering and clustering on the final reconstruction, we perform the algorithm with or without filtering and with uniform and non-uniform clustering (see Figure 8.17). In the graphs 8.15 and 8.16, the radii distributions are plotted for the

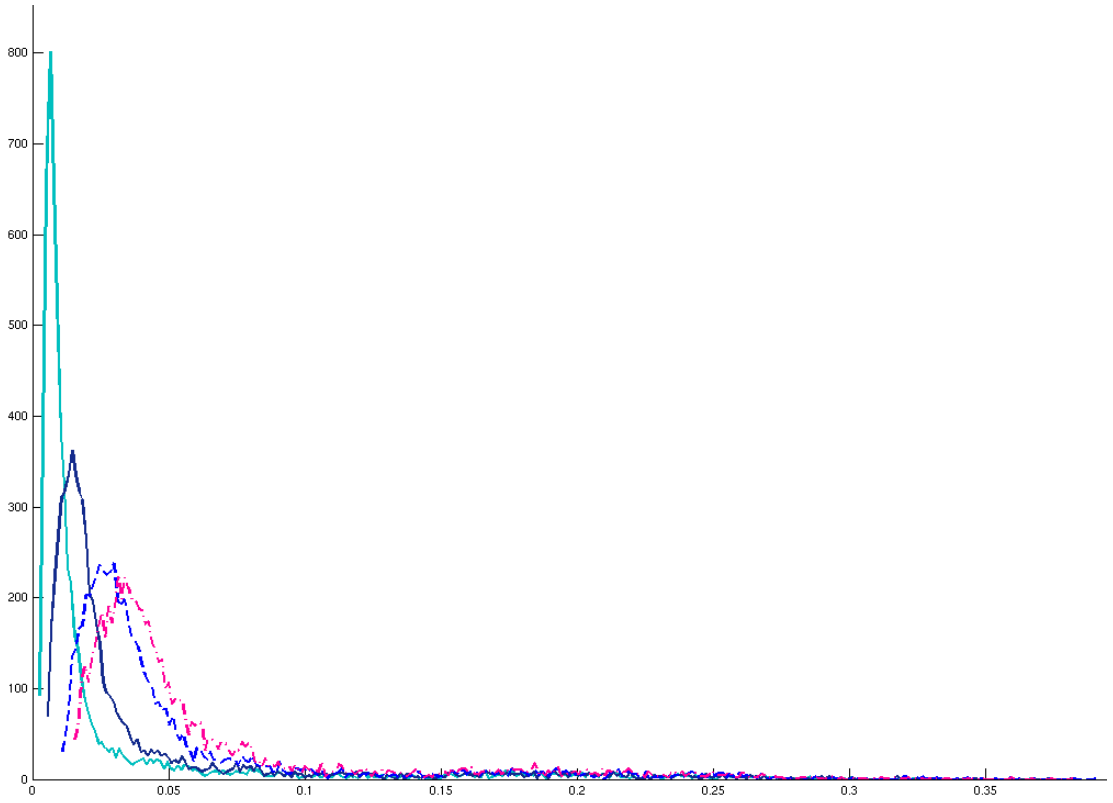


Figure 8.14: Impact of filtering over the distribution of center ball radii. The curves represent the radii distributions of 10K centers obtained by clustering with different filtering parameters (green: $\lambda = 0$, blue: $\lambda = 0.01$, doted blue: $\lambda = 0.02$, doted red: $\lambda = 0.03$). Each curve corresponds to a reconstruction shown by Figure 8.13

four cases. Note how a 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.

Uniform clustering (Figure 8.17(a) and Figure 8.17(c)) leads to a center set with a large density in the middle of the shape where the input point density is important (Figure 8.3). Conversely, a non-uniform clustering generates a center set with a better distribution on the shape (although if m is too small this leads to a disconnected shape, see Figure 8.17(b)). The filtering step handles this problem by disqualifying the centers associated to the smallest ball. The allocated budget of centers may then be used for areas with larger sf values (Figure 8.17(c)).

Noise filtering Figure 8.18 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 (Figure 8.18(a)), the λ-medial axis is stable under such perturbations, and theses

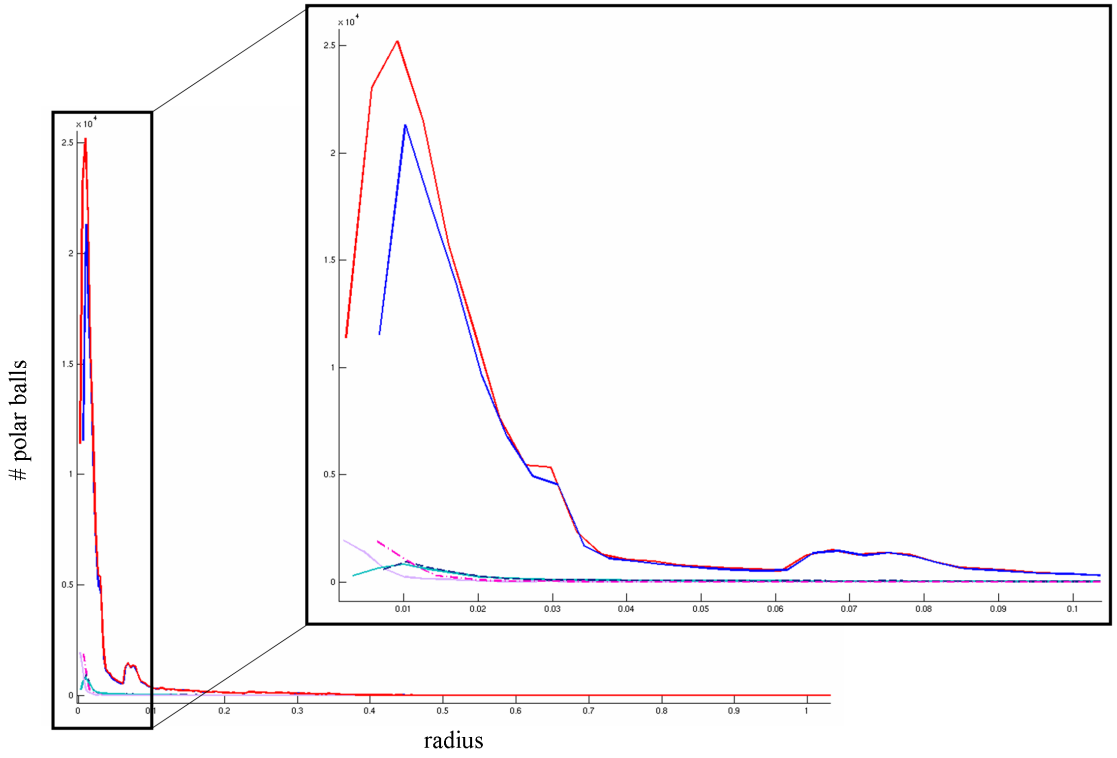


Figure 8.15: Radii distribution for the clustering/filtering approach. The Filigree Model (50k input points and 11k poles), 5k centers are selected with clustering. The pole may be filtered before the clustering.
a detail of distribution of the radii (focus on the smallest radii)
Red: all poles without filtering; Purple: uniform clustering without filtering; Green: non uniform clustering without filtering; Blue: all poles with filtering; Doted blue: uniform clustering with filtering; Doted pink: non uniform clustering with filtering.

misclassified poles are filtered out (Figure 8.18(b)).

8.4 Greedy Selection

Greedy selection allows us to select a set of centers which are well distributed, i.e. with a distribution independent from the sampling and adapted to the level of detail indirectly determined by the user-defined budget of centers. Figure 8.19 illustrates greedy center selection.

As shown by Figure 8.19(left), the allocated budget of centers is sufficient to reconstruct the knot. When the budget is not sufficient (Figure 8.19(right)), the poles corresponding to the smallest polar balls (some outside poles between the two nearby surface sheets) may not be selected. The topology of the knot is not properly reconstructed by lack of separation.

In contrast to clustering, the greedy selection promotes the largest polar balls, and the smallest polar balls are selected only if the allocated center budget is sufficient. Figure 8.20

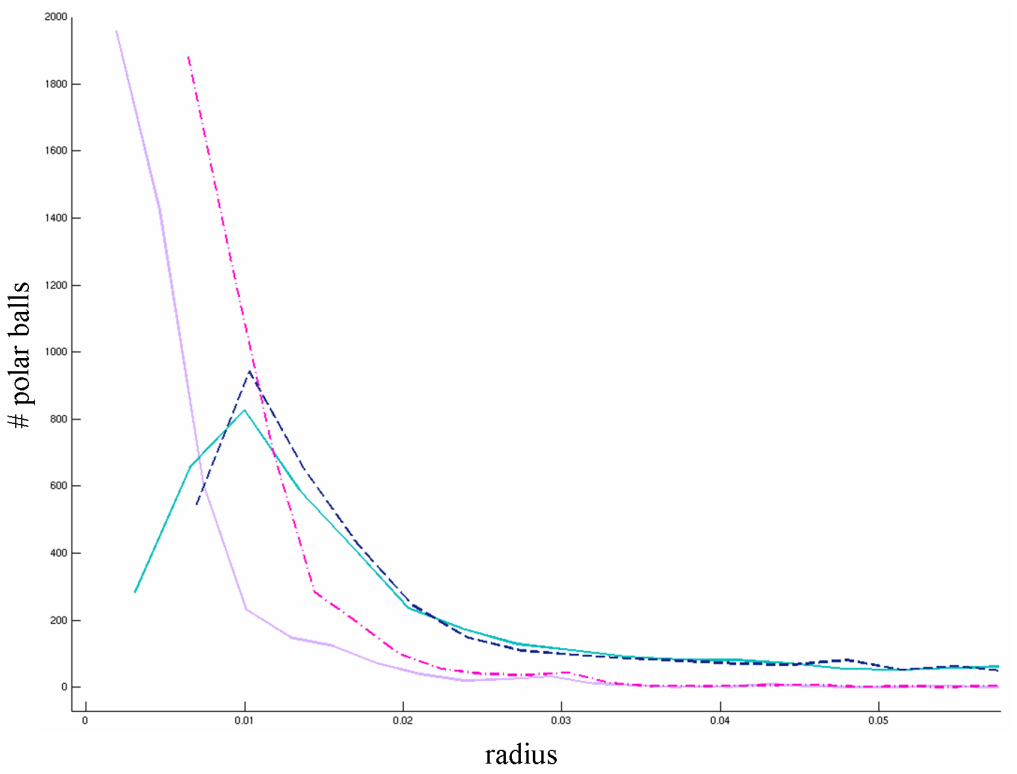


Figure 8.16: Detail of radii distribution (focus on the 4 kinds of selection). Green: non uniform clustering without filtering; Blue: all poles with filtering; Doted blue: uniform clustering with filtering; Doted pink: non uniform clustering with filtering.

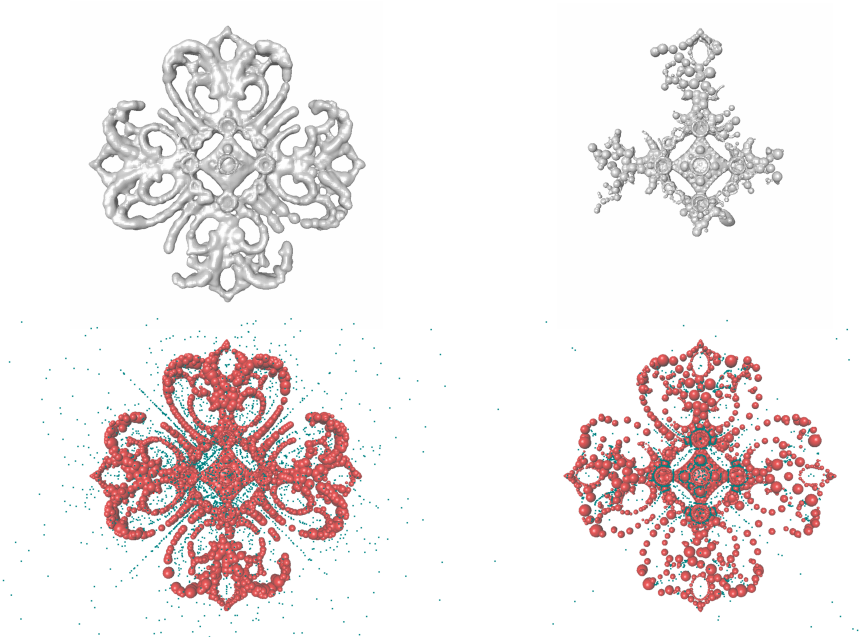
shows different reconstructions for a fixed overlapping rate threshold, $\rho = \frac{2}{3}$.

Figure 8.21 shows different reconstructions with 11K centers, with different values of ρ.

If the overlapping rate threshold ρ is large, the reconstructed surface is smooth. Indeed, the overlapping between polar balls is more important and thus produces less discontinuities. For large ρ values however, the reconstruction is less detailed. A satisfactory trade off must be manually found for each model by trial and error, and an automatic parameter selection would be desirable.

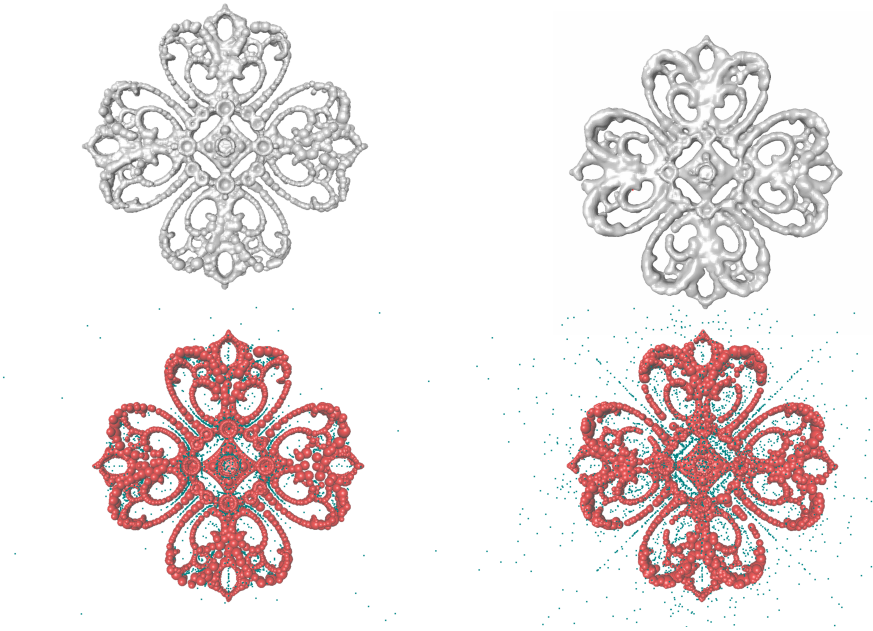
8.5 Comparing Selection Methods

The filtering-clustering approach promotes centers associated to small radius, while greedy selection selects small radius centers only if the allocated budget of centers is sufficient (see Figure 8.22). The curves 8.23 illustrate the differences between the radii distribution for the different center selection approaches.



(a) Uniform clustering

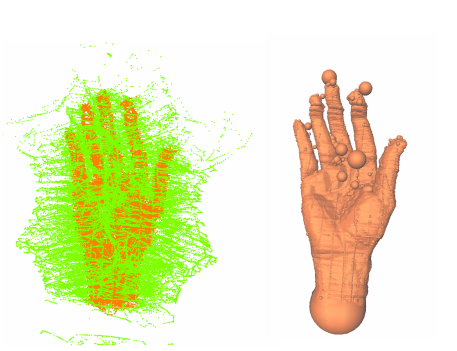
(b) Non uniform clustering



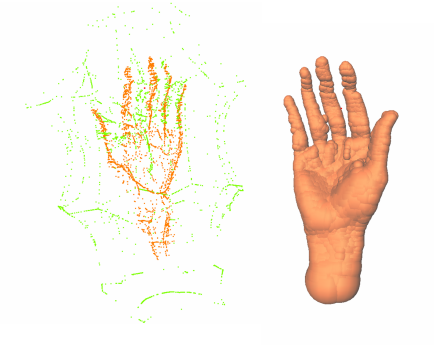
(c) Uniform clustering-filtering ($\lambda = 0.01$)

(d) Non uniform clustering-filtering($\lambda = 0.01$)

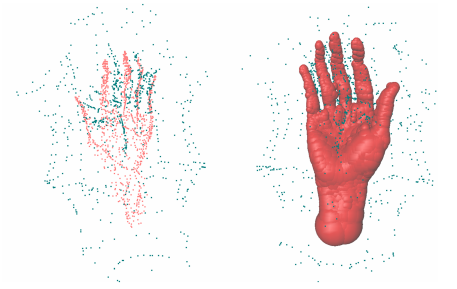
Figure 8.17: Impact of filtering and clustering. Filigree Model (50k input points and 11k poles), 5k centers are selected by clustering. The clustering step may be followed or not by filtering, and the clustering may be uniform or non-uniform. For each image the reconstructed surface is shown, as well as the set of centers with their inside polar balls (red).



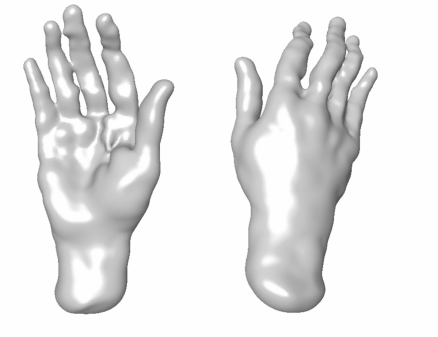
(a) orange inside poles with their polar balls and green outside poles (88K poles, some of them being misclassified);



(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 8.18: Noisy hand model (90K input points).

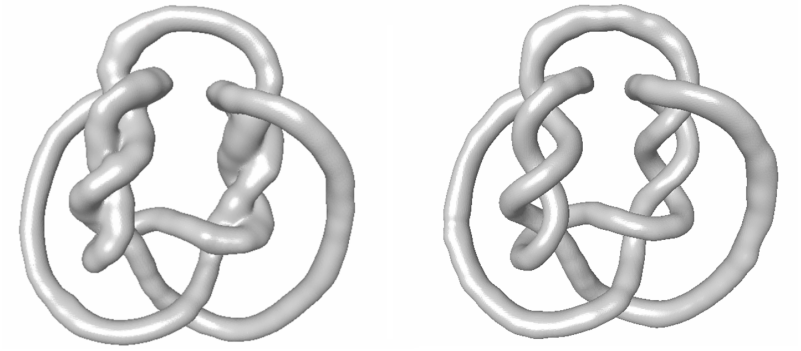
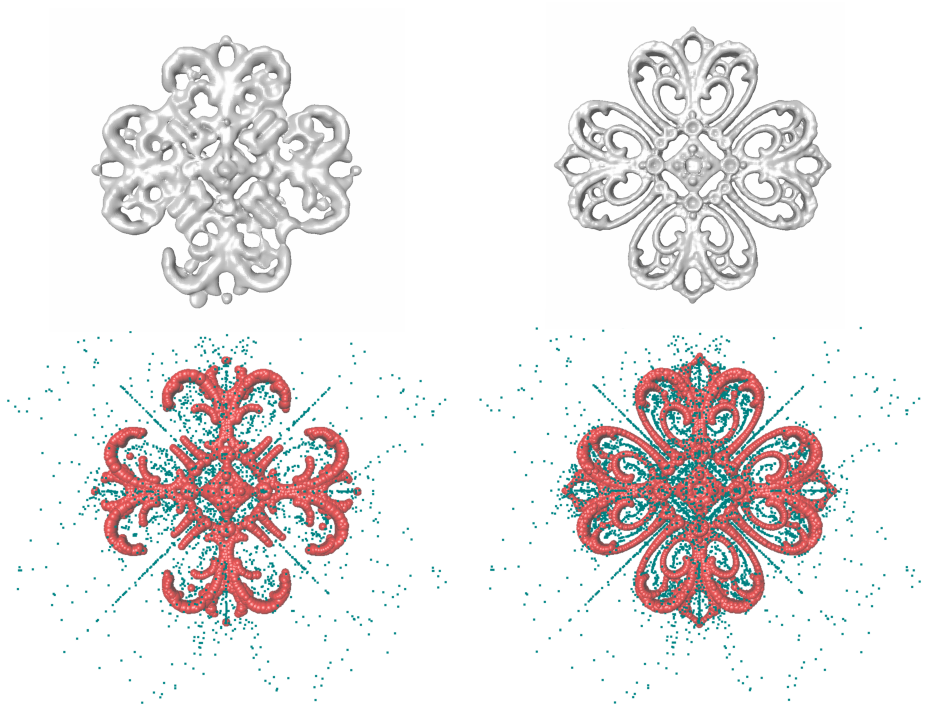
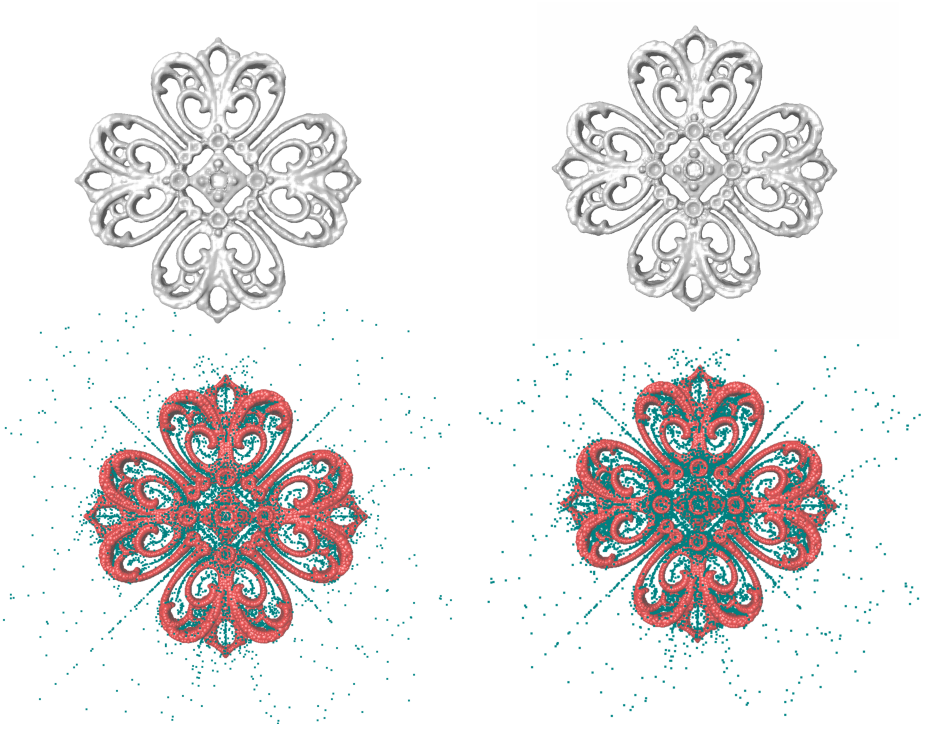


Figure 8.19: Greedy center selection ($\rho = 0.6$) on the knot model (6K input points, 12K poles). Left: Reconstruction with 1K centers. s Right: Reconstruction with 2K centers.

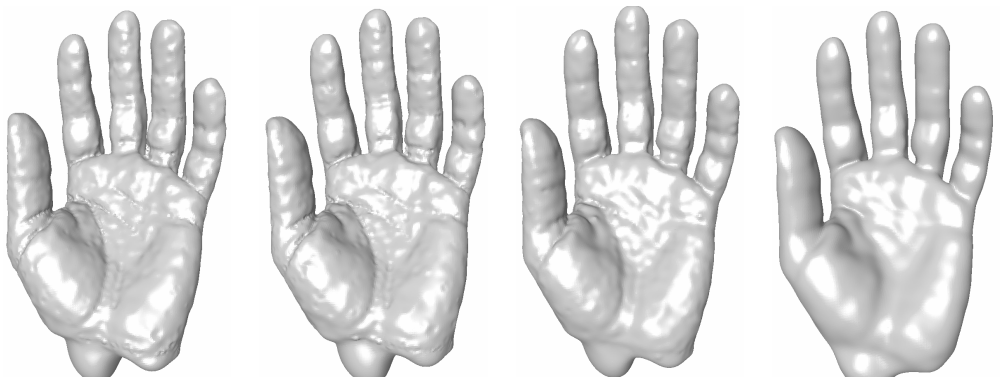


(a) $m = 5K$ (b) $m = 10K$



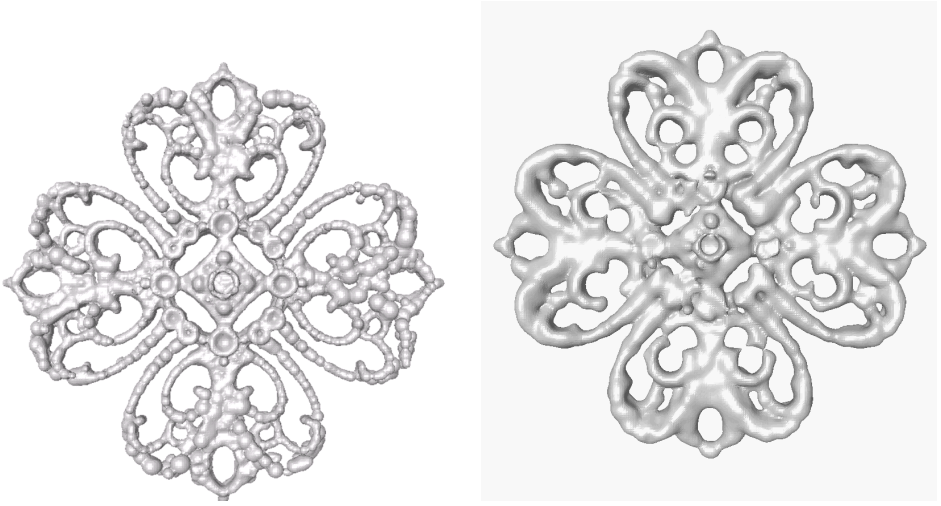
(c) $m = 15K$ (d) $m = 18K$

Figure 8.20: Greedy center selection on the Filigree model (80K input points). The overlapping rate threshold is set to $\rho = 0.6$. Top: Reconstructed surfaces. Bottom: Set of centers with their red inside polar balls.



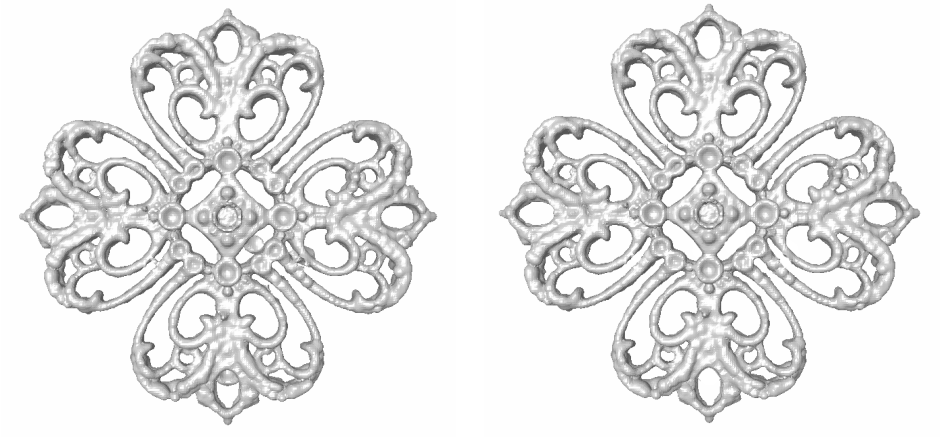
(a) $\rho = 0.1$ (b) $\rho = 0.3$ (c) $\rho = 0.6$ (d) $\rho = 0.9$

Figure 8.21: Impact of ρ over the greedy selection on the Hand model (50K input points). The number of centers is set to $10K$.



(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 8.22: Filtering-Clustering vs Greedy Selection

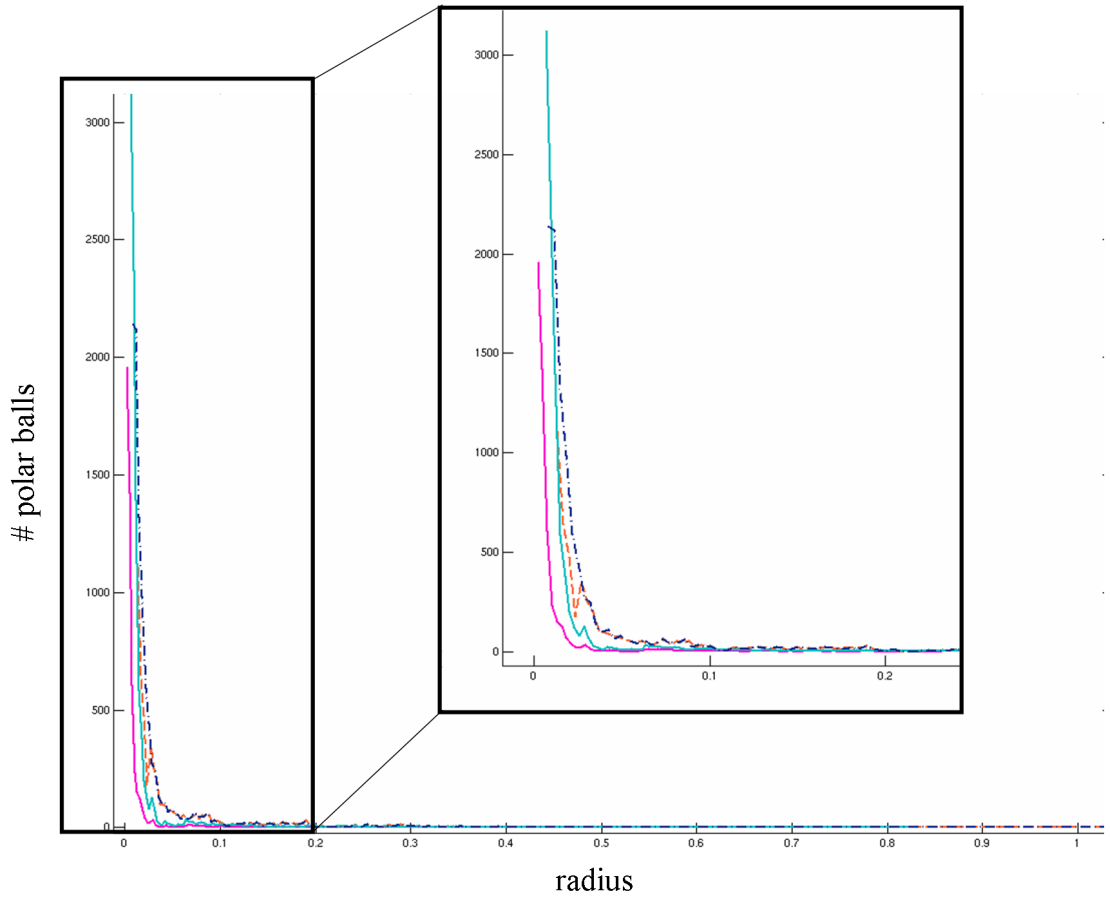


Figure 8.23: Radius distribution for Filtering-Clustering vs Greedy Selection. Filigree model (80K input points). Pink: Filtering ($\lambda = 0.01$) and clustering of 5K centers. Dotted orange: Greedy Selection ($\rho = 0.1$) of 5K centers. Green: Filtering ($\lambda = 0.01$) and clustering of 10K centers. Dotted blue: Greedy Selection ($\rho = 0.1$) of 10K centers.

Chapter 9

Conclusion and 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 Voronoi-based approaches, our method generates a smooth watertight surface by computing and contouring an implicit function, similarly to other RBF approaches. The implicit 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 some previous work.

Our approach relies on a theoretically sound framework for pole extraction and λ-medial axis filtering. In addition, this framework also provides us with reliable estimates of the normal at each data point and with an approximation of the distance to the sampled surface at each pole. As a result we can reduce the number of parameters of our algorithm to two: the number of centers and the coefficient λ, used to filter the medial axis.

In addition to filtering and clustering of the poles, we have proposed an alternate way to select the centers of the basis functions. This approach consists in a greedy pole selection based upon the overlapping ratio between polar balls.

The two methods are leading to 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 independent from the input point density.

Future Work

The only step which impairs the scalability and efficiency of our algorithm is the assembling of the final matrix. We expect to improve this aspect by an optimized implementation or

by elaborating upon different geometric data structures.

In our study the medial axis filtering process allows us to adapt the level of detail to a user-defined budget of centers, the value for λ being fixed experimentally. In the greedy selection, the parameter ρ also has to be fixed experimentally. In our experiments setting the value for ρ is more intuitive than setting the λ value, as ρ defines the maximum overlapping rate allowed. In the future, we will investigate how we can automatically adjust these parameters to accommodate for the allocated budget of centers.

Appendix

Appendix A

Voronoi, Delaunay

A.1 Voronoi Diagram

Definition A.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.*

$$V(p, P) = \{x \in \Omega : \forall q \in P \|x - p\| \leq \|x - q\|\}. \tag{A.1}$$

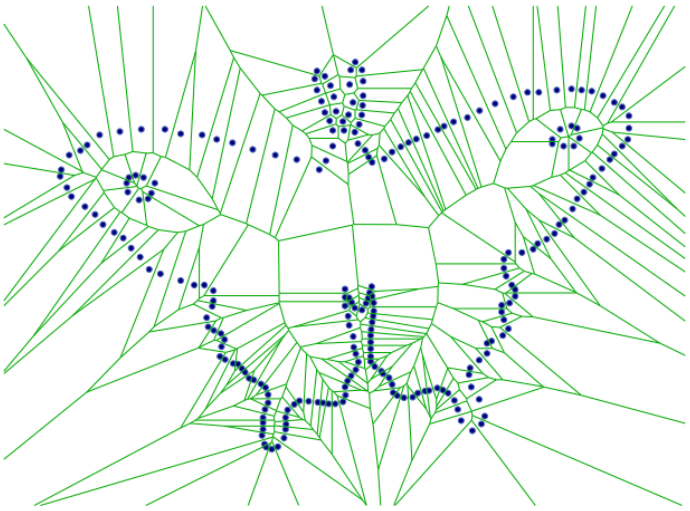


Figure A.1: 2D Voronoi diagram

A.2 Delaunay Triangulation

The dual of the Voronoi diagram $Vor(P)$ is called the Delaunay triangulation $Del(P)$.

Definition A.2. *Delaunay triangulation:*
Whenever a collection $V_1 \dots V_k$ of Voronoi cells, corresponding to points $p_1 \dots p_k$, has a non-empty intersection, the simplex whose vertices are $p_1 \dots p_k$ 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 A.2 for a 2D example of a Delaunay triangulation.

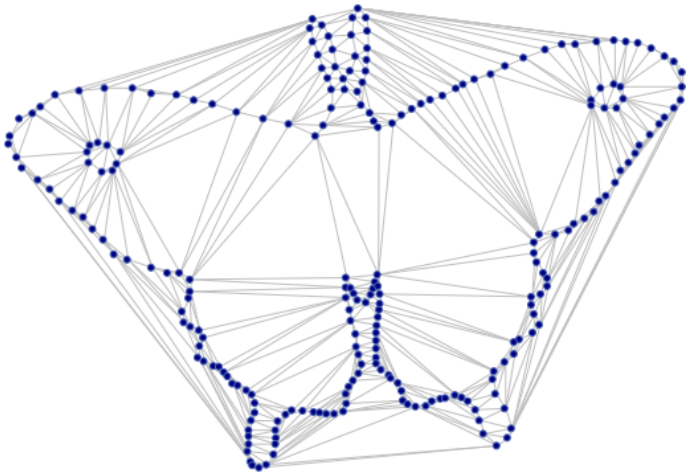


Figure A.2: 2D Delaunay triangulation

A.3 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*,

$$\pi_{(z,r)} : \mathbb{R}^3 \to \mathbb{R}, x \longmapsto \|x - z\|^2 - r \tag{A.2}$$

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.

Appendix B

Medial Axis and Local Feature Size

B.1 Medial Axis

Definition B.1. *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.B.1).

Definition B.2. *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.B.1).

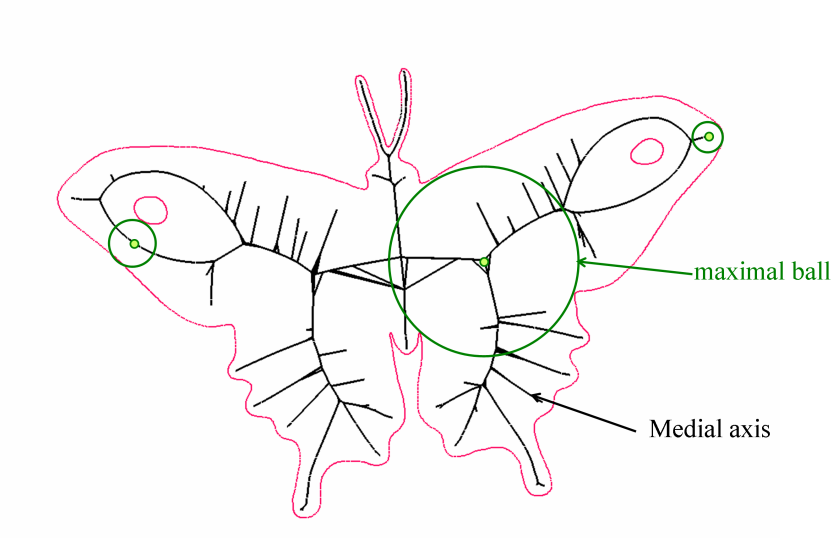


Figure B.1: Inside Medial Axis. A 2D shape (red) and its inside medial axis (black).

A profusion of methods have been proposed to extract the medial axis. The exact computation of the medial axis is difficult in the general case. 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 in the centers of maximal spheres included in $\mathbb{R}^3 \backslash S$. Here maximal mean with respect to inclusion among spheres. For a smooth surface S the closure of the medial axis is actually equal to the skeleton of $\mathbb{R}^3 \backslash S$.

B.2 Local Feature Size

The local feature size is a function $lfs\colon S \to R$ 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 a 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 with respect 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.

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.

[ACA07] R. Allègre, R. Chaine, and S. Akkouche. A flexible framework for surface reconstruction from large point sets. *Computers and Graphics*, 31(2):190–204, 2007.

[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.

[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.

[AM97] Dominique Attali and Annick Montanvert. Computing and simplifying 2d and 3d continuous skeletons. *Computer Vision and Image Understanding*, 67(3):261–273, September 1997.

[AMTI96] D. Attali, A. Montanvert, and L.T. TIMC-IMAG. Modeling noise for a better simplification of skeletons. *Image Processing, 1996. Proceedings., International Conference on*, 3, 1996.

[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] J. Bloomenthal. An implicit surface polygonizer. *Graphics Gems IV*, 1:324–349, 1994.

[BO05] J.D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, 2005.

[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] JC Carr, RK Beatson, JB Cherrie, TJ Mitchell, WR Fright, BC McCallum, and TR Evans. Reconstruction and representation of 3D objects with radial basis functions. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, 2001.

[CFB97] JC Carr, WR Fright, and RK Beatson. Surface interpolation with radial basis functions for medicalimaging. *Medical Imaging, IEEE Transactions on*, 16(1):96–107, 1997.

[CGY04] Frédéric Cazals, Joachim Giesen, and Mariette Yvinec. Delaunay triangulation based surface reconstruction: a short survey. Research Report 5394, INRIA, 2004.

[Cha03] Raphaëlle Chaine. A geometric convection approach of 3-d reconstruction. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 218–229, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. *Computer Graphics*, 30(Annual Conference Series):303–312, 1996.

[CL05] F. Chazal and A. Lieutier. The "λ-medial axis". *Graphical Models*, 67(4):304–331, 2005.

[CP05] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005.

[CSM03] David Cohen-Steiner and Jean-Marie Morvan. Approximation of the curvature measures of a smooth surface endowed with a mesh. Research Report 4867, INRIA, 2003.

[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.

[DR95] N. Dyn and A. Ron. Radial basis function approximation: from gridded centers to scattered centers. *Proc. London Math. Soc*, 71(3):76–108, 1995.

[DS05] Tamal K. Dey and Jian Sun. An adaptive mls surface for reconstruction with guarantees. In *Symposium on Geometry Processing*, pages 43–52, 2005.

[DTS01] H.Q. Dinh, G. Turk, and G. Slabaugh. Reconstructing surfaces using anisotropic basis functions. pages 606–613, 2001.

[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.

[Ede92] H. Edelsbrunner. *Weighted Alpha Shapes*. University of Illinois at Urbana-Champaign, Dept. of Computer Science, 1992.

[EM94] Herbert Edelsbrunner and Ernst P. Mücke. 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.

[GO93] Ardeshir Goshtasby and William D. O'Neill. Surface fitting to scattered data by a sum of gaussians. *Comput. Aided Geom. Des.*, 10(2):143–156, 1993.

[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.

[JWB+06] Philipp Jenke, Michael Wand, Martin Bokeloh, Andreas Schilling, and Wolfgang Strasser. Bayesian point cloud reconstruction. In *Proceedings EUROGRAPHICS '06*, volume 25, 2006.

[KB04] L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004.

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Fourth Eurographics Symposium on Geometry Processing*, pages 61–70, June 2006.

[Kir00] M. Kirby. *Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns*. John Wiley & Sons, Inc. New York, NY, USA, 2000.

[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. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[LBC96] Z. Lei, Michael M. Blane, and David B. Cooper. 3L fitting of higher degree implicit polynomials. In *Proceedings of Third IEEE Workshop on Applications of Computer Vision*, Sarasota, Florida, USA, 1996.

[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.

[MYC+01] B.S. Morse, T.S. Yoo, D.T. Chen, P. Rheingans, and K. R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *SMI '01: Proceedings of the International Conference on Shape Modeling & Applications*, page 89. IEEE Computer Society, 2001.

[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.

[OBS05] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Multi-scale and adaptive cs-rbfs for shape reconstruction from cloud of points. In Neil A. Dodgson, Michael S. Floater, and Malcolm A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 143–154. Springer, Berlin, Germany, 2005.

[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.* Massachusetts Institute of Technology, Artificial Intelligence Laboratory and Center for Biological Information Processing, Whitaker College, 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.

[Pow87] MJD Powell. Radial basis functions for multivariable interpolation: a review. *Clarendon Press Institute Of Mathematics And Its Applications Conference Series*, pages 143–167, 1987.

[Pow90] M. Powell. The Theory of Radial Basis Function Approximation in 1990. *Advances in numerical analysis. Vol. 2: Wavelets, subdivision algorithms, and radial basis functions, Proc. 4th Summer Sch., Lancaster/UK*, page 2, 1990.

[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.

[RJT+05] Patrick Reuter, Pierre Joyot, Jean Trunzler, Tamy Boubekeur, and Christophe Schlick. Surface reconstruction with enriched reproducing kernel particle approximation. In *IEEE/Eurographics Symposium on Point-Based Graphics*, pages 79–87. Eurographics/IEEE Computer Society, 2005.

[RY06] Laurent Rineau and Mariette Yvinec. 3d surface mesher. In CGAL Editorial Board, editor, *CGAL-3.2 User and Reference Manual*. 2006.

[SBS06] Oliver Schall, Alexander Belyaev, and Hans-Peter Seidel. Adaptive fourier-based surface reconstruction. In M.-S. Kim and K. Shimada, editors, *Geometric Modeling and Processing*, volume 4077 of *Lecture Notes in Computer Science*, pages 34–44, Pittsburgh, Pennsylvania, USA, 2006. Springer.

[Sch95] R. Schaback. Creating surfaces from scattered data using radial basis functions. *Mathematical Methods for Curves and Surfaces*, pages 477–496, 1995.

[SLS+06] Andrei Sharf, Thomas Lewiner, Ariel Shamir, Leif Kobbelt, and Daniel Cohen-Or. Competing fronts for coarse–to–fine surface reconstruction. In *Eurographics*, pages 389–398, Vienna, september 2006.

[Tau94] Gabriel Taubin. Distance approximations for rasterizing implicit curves. *ACM Trans. Graph.*, 13(1):3–42, 1994.

[TBC95] Nicolas Tsingos, Eric Bittar, and Marie-Paule Cani. Semi-automatic reconstruction of implicit surfaces for medical applications. In *Computer Graphics International*, june 1995. Published under the name Marie-Paule Gascuel.

[TI04] Schlick 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 New York, 2004.

[Wu95] Z. Wu. Compactly supported positive definite radial functions. *Advances in Computational Mathematics*, 4:283.292, 1995.

[ZOF01] H.K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. *1st IEEE Workshop on Variational and Level Set Methods, 8th ICCV*, 80(3):194–202, 2001.

Abstract

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.

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 levels of detail and obtain a more compact final representation.

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 to obtain a more compact representation and to reduce the evaluation cost. Reducing the number of basis function is equivalent to reducing the number of points (*centers*) where the functions are centered. Our goal consist in selecting a "little" set of relevant centers, to reduce the number of centers while maintaining decent fitting accuracy. We depart from previous work by relaxing the one-to-one correspondence between the centers and the data points. We use as centers of the 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 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.

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.