Robust shape reconstruction from defect-laden data

Simon Giraudot

UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS

ÉCOLE DOCTORALE STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE

pour l'obtention du grade de

Docteur en Sciences

de l'Université de Nice-Sophia Antipolis

Mention informatique

présentée et soutenue par

Simon GIRAUDOT

Reconstruction robuste de formes à partir de données imparfaites

(Robust Shape Reconstruction from Defect-Laden Data)

Thèse dirigée par Pierre ALLIEZ
soutenue le 22 mai 2015

Jury :

Rapporteurs :	Niloy MITRA	Professeur, University College London
	Marco ATTENE	Chargé de recherche, IMATI (Gênes)
Examinateurs :	Gaël GUENNEBAUD	Chargé de recherche, Inria Bordeaux
	Dominique ATTALI	Directeur de recherche CNRS, Gipsa-Lab (Grenoble)
	Stéphane NULLANS	Ingénieur R&D, Autodesk (Sophia Antipolis)
Invité :	David COHEN-STEINER	Chargé de recherche, Inria Sophia Antipolis
Directeur :	Pierre ALLIEZ	Directeur de recherche, Inria Sophia Antipolis

Abstract

Surface reconstruction from point clouds is a widely explored problem in geometry processing. Although a high number of reliable methods have been developped over the last two decades, they often require additional measurements such as normal vectors or visibility information. Furthermore, robustness to defect-laden data is only achieved through strong assumptions and remains a scientific challenge.

We focus on defect-laden, unoriented point clouds and propose two new reconstruction approaches designed for two specific classes of output surfaces.

The first method is noise-adaptive and specialized to smooth, closed shapes. It takes as input a point cloud with variable noise and outliers, and comprises three main steps. First, we compute a novel noise-adaptive distance function to the inferred shape, which relies on the assumption that the inferred shape is a smooth submanifold of known dimension. Second, we estimate the sign and confidence of the function at a set of seed points, through minimizing a quadratic energy expressed on the edges of a uniform random graph. Third, we compute a signed implicit function through a random walker approach with soft constraints chosen as the most confident seed points computed in previous step.

The second method generates piecewise-planar surfaces, possibly non-manifold, represented by compact triangle meshes. Through multiscale region growing of Hausdorff-error-bounded convex planar primitives, we infer both shape and connectivity of the input and generate a simplicial complex that efficiently captures large flat regions as well as small features and boundaries. Imposing convexity of primitives is shown to be crucial to both the robustness and efficacy of our approach. We provide a variety of results from both synthetic and real point clouds.

Introduction (français)

Au cours des vingt dernières années, de nombreux algorithmes de reconstruction de surface ont été développés. Néanmoins, des données additionnelles telles que les normales orientées sont souvent requises et la robustesse aux données imparfaites est encore un vrai défi.

Dans cette thèse, nous traitons de nuages de points non-orientés et imparfaits, et proposons deux nouvelles méthodes gérant deux différents types de surfaces.

La première méthode, adaptée au bruit, s'applique aux surfaces lisses et fermées. Elle prend en entrée un nuage de points avec du bruit variable et des données aberrantes, et comporte trois grandes étapes. Premièrement, en supposant que la surface est lisse et de dimension connue, nous calculons une fonction distance adaptée au bruit. Puis nous estimons le signe et l'incertitude de la fonction sur un ensemble de points-sources, en minimisant une énergie quadratique exprimée sur les arêtes d'un graphe uniforme aléatoire. Enfin, nous calculons une fonction implicite signée par une approche dite *random walker* avec des contraintes molles choisies aux points-sources de faible incertitude.

La seconde méthode génère des surfaces planaires par morceaux, potentiellement non-variétés, représentées par des maillages triangulaires simples. En faisant croitre des primitives planaires convexes sous une erreur de Hausdorff bornée, nous déduisons à la fois la surface et sa connectivité et générons un complexe simplicial qui représente efficacement les grandes régions planaires, les petits éléments et les bords. La convexité des primitives est essentielle pour la robustesse et l'efficacité de notre approche.

Résumé (français)

Nous présentons deux nouveaux algorithmes de reconstruction de formes à partir de nuages de points bruts. Le premier est adaptatif au bruit et s'applique aux surfaces lisses fermées, tandis que le second produit des surfaces planes par morceaux avec bords en décomposant le nuage de points en primitives convexes et planes.

Reconstruction adaptative au bruit

Fonction distance adaptative au bruit

Mullen et al. [64] ont proposé un algorithme de reconstruction de forme basé sur cette fonction distance robuste [23] :

$$d^2_{\mu,m_0}(\boldsymbol{x}) = \frac{1}{m_0}\int_{B(\boldsymbol{x},r_{\mu,m_0}(\boldsymbol{x}))} \|\boldsymbol{x}-\boldsymbol{y}\|^2 \mathrm{d}\mu(\boldsymbol{y}).$$

où μ est une mesure de second moment fini, i.e., avec un support fini sur son domaine de définition, w est un point de requête et m_0 est un paramètre d'échelle.

Nous étudions le comportement de cette fonction dans deux cas particuliers. Dans le cas d'un bruit ambiant dans un espace de dimension d, cette fonction se simplifie :

$$d^2_{\mu,m_0} = c_1 m_0^{2/d}.$$

Pour un point de requête fixé, d_{μ,m_0} est proportionnel à $m_0^{1/d}$. Pour un paramètre $\alpha > 1/d$, $\frac{d_{\mu,m_0}}{m_0^\alpha}$ est donc décroissant avec m_0. Cette fonction atteint son minimum quand toute la masse de μ est incluse dans m_0.

Considérons maintenant le cas d'une mesure uniforme sur un sous-espace de dimension

k dans un espace de dimension d. La fonction distance dépend de la distance h entre un point de requête $\boldsymbol{x}$ et ce sous-espace :

$$d^2_{\mu,m_0} = h^2 + c_2 m_0^{2/k}.$$

Pour un point de requête fixé, d_{μ,m_0} est proportionnel à $m_0^{1/k}$. Pour un paramètre $\alpha < 1/k$, $\frac{d_{\mu,m_0}}{m_0^\alpha}$ est donc croissant avec m_0.

Un cas réaliste est un mélange de ces deux cas particuliers :

- lorsque l'échelle m est plus faible que la dimension apparente, $\frac{d_{\mu,m}}{m^\alpha}$ se comporte de la même manière que dans un bruit ambiant et est décroissant avec m ;
- lorsque l'échelle m est plus grande que la dimension apparente, $\frac{d_{\mu,m}}{m^\alpha}$ se comporte de la même manière qu'avec un sous-espace de dimension k et est croissant avec m.

L'échelle m où $\frac{d_{\mu,m}}{m^\alpha}$ atteint son minimum est l'échelle locale, et $\frac{d_{\mu,m}}{m^\alpha}$ est alors la valeur de la fonction adaptative au bruit. Plus précisément, cette fonction adaptative est définie de cette manière :

$$\delta_{\mu,\alpha} = \inf_{m_0>0} \frac{d_{\mu,m_0}}{m_0^\alpha},$$

où α est un paramètre dépendant uniquement de la dimension (3/4 pour une courbe en 2D ou 5/12 pour une surface en 3D).

La fonction $d_{\mu,m}$ étant $\frac{1}{\sqrt{m_O}}$-robuste, cette fonction $\delta_{\mu,\alpha}$ conserve les propriétés robustesse à condition d'imposer une borne inférieure à m_0 (6 points en pratique). Elle est calculée rapidement à l'aide d'une approche multiéchelle et est stockée sur une triangulation de Delaunay adaptée.

Suppositions de signes

Afin de signer la fonction adaptative au bruit, il est nécessaire d'effectuer des suppositions de signes. Nous proposons une généralisation de la méthode consistant à tirer des rayons en utilisant des segments.

Considérons un segment entre deux points arbitraires de l'espace. Pour estimer s'ils sont du même côté de la forme (à l'extérieur ou à l'intérieur), il faut calculer la parité du nombre de fois que la forme est croisée. Les valeurs de la fonction adaptative dépendant

du niveau local de bruit, il est impossible d'utiliser un seuil. Au lieu de cela, nous utilisons l'hypothèse que la fonction signée doit être la plus lisse possible : en essayant d'inverser le sens de variation de la fonction à chaque minimum local le long d'un segment, nous choisissons la solution la plus lisse.

Ces hypothèses de signe sont locales et relative. Afin d'obtenir une solution globale et absolue, nous utilisons un graphe définie sur une grille régulière couvrant l'espace. Des arêtes (i, j) sont tirées aléatoirement et signées avec la méthode décrite ci-dessus. L'énergie suivante est minimisée :

$$E_G(f) = \sum_{(i,j)\in G} (s_i + \varepsilon_{i,j} s_j)^2,$$

où $\varepsilon_{i,j}$ sont les hypothèses de signe sur les arêtes et s_i les signes des nœuds du graphe. Un solveur linéaire permet de minimiser $E_G(f)$ et de signer les nœuds du graphe.

Signer la fonction distance

Grâce aux suppositions de signes calculées précédemment, nous signons la fonction adaptative $\delta_{\mu,\alpha}$. Nous utilisons une approche de *marche aléatoire* ([40]) afin de minimiser l'énergie suivante :

$$E_{g,T} = \int_\Omega w(x)|\nabla g(x)|^2 \mathrm{d}x,$$

avec g la fonction implicite signée et

$$w(x) = \begin{cases} \delta_{\mu,\alpha}(x) & \text{si } K(x) < K_{max} \\ +\infty & \text{sinon} \end{cases}$$

Cette énergie est également minimisée grâce à un solveur linéaire.

Résultats & conclusion

Les résultats sont visibles Section 5.4. Cette méthode ne nécessite aucune information supplémentaire (vecteurs normaux...), est robuste au bruit, aux données aberrantes, aux données manquantes et s'adapte automatiquement à de multiples niveaux de bruits. Elle passe à l'échelle, ne nécessitant que deux solveurs linéaires sur des matrices creuses.

Les deux contributions principales sont la fonction adaptative au bruit et la méthode de signature de la fonction basée sur des segments.

Reconstruction plane par morceaux

Approximation

Cette seconde méthode utilise une approximation du nuage d'entrée à l'aide de primitives convexes planes. Une primitive Π_i est définie par :

- son sous-ensemble d'*inliers* $\mathcal{P}_i$;
- son plan support S_i calculé par *analyse en composante principale* de $\mathcal{P}_i$;
- l'enveloppe convexe 2D des projections de $\mathcal{P}_i$ sur S_i.

Afin de quantifier la qualité de l'approximation offerte par ces primitives, nous utilisons une distance de Hausdorff symétrique :

$$d_{\mathrm{H}}(\Pi_i, \mathcal{P}_i) = \max\{ \sup_{a \in \Pi_i} \left[\inf_{b \in \mathcal{P}_i} d(a,b) \right], \sup_{b \in \mathcal{P}_i} \left[\inf_{a \in \Pi_i} d(a,b) \right] \},$$

où $d(a,b)$ est la distance euclidienne. Cette métrique peut-être décomposée en deux :

- $H(\mathcal{P}_i, \Pi_i) = \sup_{b \in \mathcal{P}_i} \inf_{a \in \Pi_i} d(a,b)$ est la distance maximale depuis $\mathcal{P}_i$ vers sa primitive associée : elle mesure un niveau de bruit ;
- $H(\Pi_i, \mathcal{P}_i) = \sup_{a \in \Pi_i} \inf_{b \in \mathcal{P}_i} d(a,b)$ est la distance maximale depuis une primitive vers ses *inliers* : elle est liée à la densité d'échantillonnage.

Cette distance de Hausdorff est calculée par division récursive de la triangulation de Delaunay d'une primitive.

Algorithme grossier vers fin

Nous proposons un algorithme de construction de ces primitives efficace du grossier vers le fin. Le nuage de points d'entrée est pré-traité avec un algorithme de partition hiérarchique [68] : plusieurs échelles sont stockées, l'indice $i = 0$ correspond à l'échelle la plus fine (nuage d'entrée). Chaque point d'une échelle supérieure est associé à une mesure de planarité basé sur l'analyse en composante principale de son sous-ensemble de points :

$$sv(c) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}.$$

Construction de primitives

L'algorithme de construction des primitives peut être résumé de cette manière :

1. initialiser Π_i sur l'échelle non-vide la plus grossière ;
2. aller à l'échelle immédiatement plus fine ;
3. trouver les *inliers* du plan support de Π_i à cette échelle ;
4. faire grossir Π_i en se basant sur ces *inliers* ;
5. retirer tous les inliers de Π_i trouvé à l'échelle courante ;
6. revenir à l'étape 2 s'il reste des points.

Détection d'adjacences

Les adjacences entre primitives sont détectées en échantillonnant leurs bords et en insérant des coins lorsque des échantillons deviennent proches d'autres primitives. Afin de gérer les cas non-variétés, des échantillons additionnels peuvent être insérés à l'intérieur des primitives.

Les coins sont ensuite fusionnés s'ils sont plus proches que la tolérance d'échantillonnage donnée. Les positions des sommets obtenus sont optimisés au sens des moindres carrés avec les bords des primitives.

Maillage

Le maillage des primitives est réalisé à l'aide de triangulation de Delaunay contraintes 2D. Les bords des primitives deviennent des bords contraints, et les cas non-variétés sont insérés comme sommets internes et comme arêtes internes contraintes.

Résultats & conclusion

Les résultats sont donnés Section 9.7. Cette méthode est robuste au bruit et à un échantillonnage non-uniforme, tant que ces deux défauts restent en-dessous des tolérances Hausdorff données.

Nos contributions sont un algorithme d'approximation de nuage de point avec bornes d'erreur globales garanties ainsi qu'un algorithme de reconstruction basé sur cette décomposition convexe efficace du nuage de points d'entrée.

Notations

Input

d	Dimension of the space (in our context, $d = 2$ or 3).
k	Dimension of the inferred shape (in our context, $d = 1$ or 2).
n	Number of points in the point cloud.
$\mathcal{P} = \{\boldsymbol{p_i}\}_{i \in [1:n]}$	Point cloud.
$\mathcal{P}_i$	Point subset.
$\mathcal{N} = \{\boldsymbol{n_i}\}_{i \in [1:n]}$	Set of normal vectors ($\boldsymbol{p_i}$ has normal vector $\boldsymbol{n_i}$).
$\mathcal{V} = \{\boldsymbol{v_i}\}_{i \in [1:n]}$	Set of visibility vector ($\boldsymbol{p_i}$ has visibility vector $\boldsymbol{v_i}$).
μ	Discrete measure

Scale

s	Global scale.
$s(q)$	Local scale at query point q.
m_0	Global scale in the sense of a subset of distribution μ.
$m(q)$	Local scale in the sense of a subset of distribution μ at query point q.
K	Global scale in the sense of a number of nearest neighbors.
$K(q)$	Local scale in the sense of a number of nearest neighbors at query point q.
$N_K(q)$	Set of K nearest neighbors of query point q.
ε_S	Sampling tolerance.
ε_N	Noise level tolerance.

Data structures

$\mathrm{CH}(\mathcal{P})$	Convex hull of $\mathcal{P}$.
$\mathrm{Del}(\mathcal{P})$	Delaunay triangulation of $\mathcal{P}$.
$\mathrm{Vor}(\mathcal{P})$	Voronoi diagram of $\mathcal{P}$.
Π_i	Primitive associated to point subset $\mathcal{P}_i$.

Contents

1 Introduction

1.1 Shape Reconstruction

1.1.1 Motivation

Initially tackled in computational geometry, the shape reconstruction problem has spread to a wider diversity of communities over the last two decades. As the number of applications increases, so do the scientific challenges raised by the *digitization of the world*.

One motivating application depicted by Figure 1.1 is *cultural heritage*: scanning and reconstructing pieces of art, antiques or historical monuments, for the sake of archives and collective memory. For this class of application, the reconstruction pipeline must be accurate and reliable, the same way a photography should provide a faithful representation of a painting.

The scope of shape reconstruction is wider than archiving. It also significantly helps *reverse engineering* processes, where the quality of the reconstruction is a strong requirement. Such quality requirement becomes crucial for *medical simulation*, for example with patient-specific anatomical modeling.

Finally, the *automatic modeling of urban scenes* from acquired geometric data sets is

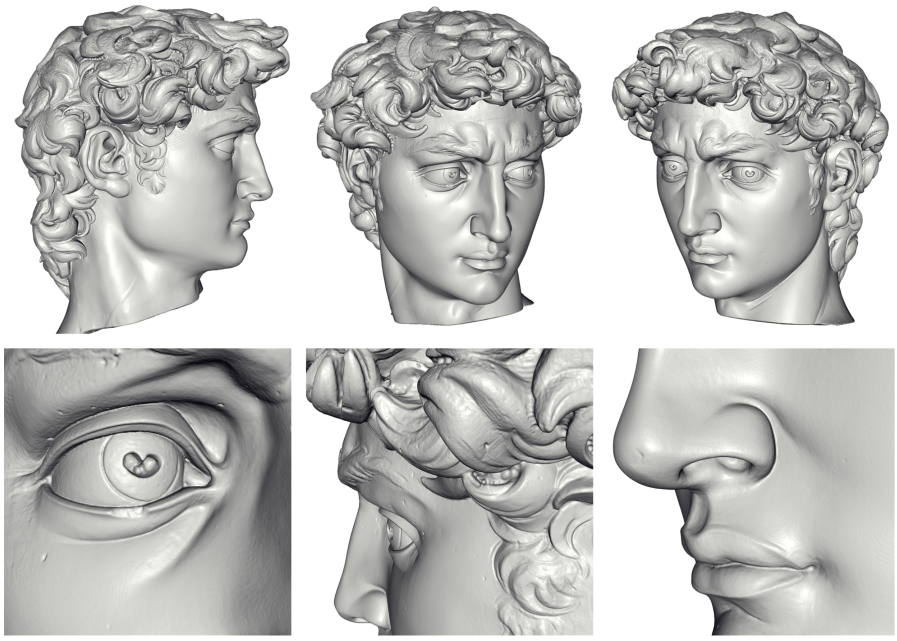


Figure 1.1: **Cultural heritage** [48]**.** Reconstruction of Michelangelo's David.

used for an increasingly wide range of applications. For example, *Google Earth* has included for a few versions now automatically reconstructed 3D models of some big cities as shown on Figure 1.2.

Shape reconstruction is still a scientific challenge as it raises a number of open problems relating to geometric modeling and information processing. The problem of converting discrete samples to a continuous shape cannot be handled directly and has to be considered with specific *a priori* and assumptions. Furthermore, the need for efficient shape reconstruction dealing with massive data sets grows stronger.

Finally, it is still a topic of interest as it relates to many other scientific challenges, such as shape detection, dimension and topology detection, object classification and information theory.

1.1.2 Wide Context

The literature on the topic is wide. The keywords "surface reconstruction" are found in the title of around $7,300$ articles indexed on *Google Scholar*. This is a consequence of the wide

Figure 1.2: **Google Earth.** Screenshot from the *Google Earth* software of a reconstructed scan of Place Massena in Nice, France.

range diversity of use cases: from organic to man-made point clouds, through architectural, urban or indoor environment and computer-aided design (CAD).

The range of defects in the measurement data is also wide: noise, outliers and non-uniform sampling. Solutions to handle one or several of these defects have been contributed in the past decade often at the cost of some additional requirements: normal vectors, visibility information or color attributes.

The output reconstruction may also have to fulfill a set of specific requirements: cultural heritage requires high fidelity to the scanned object, whereas large scale urban scenes should be reconstructed with some level of abstraction in the form of a low number of primitives. If reconstruction is a first step for medical simulation, emphasis is put on the faithfulness of to the geometric properties such as topology or watertightness.

1.1.3 New Context

Over the last few years, the context of shape reconstruction has changed. Although sensors are constantly evolving and technical advances are regularly made in acquisition devices, the proliferation of low-cost various sources has led to novel challenges.

The *Microsoft Kinect* is a typical example: a cheap device accessible to general audience (part of a video game console) that acquires point clouds with very particular defects. These point clouds contain for instance noise depending on the materials of the scanned objects,

and growing rapidly with depth.

Point clouds acquired from various sensors can be merged to produce a single larger point cloud, resulting in scenes with heterogeneous defects and major registration problems. *Super-resolution* is the research field that tackles these issues.

With the booming of social networks, community data is also becoming a facet of the problem that cannot be ignored, providing huge databases of point clouds with no control on the acquisition process.

This new context also brings new directions to shape reconstruction. Because of the loss of control of the acquisition process and the growth of disseminated sensor, interest for online reconstruction is increasing. For example, efficiently handling sensor networks may turn out to be crucial for helping authorities during natural disasters: there is a need of up-to-date and quickly reconstructed data.

The time require to perform reconstruction is expected to decrease even if the complexity of the point clouds grows. At the same time, the increasing variety of sensors and of acquired scenes has lead to a need for a higher robustness to defects with as little requirements on the input as possible.

1.2 Input

1.2.1 Sensors

The input of shape reconstruction is usually a point cloud, possibly provided with additional information as we shall discuss later.

The main target of shape reconstruction is digitizing the real world: the input point clouds are not generated but *acquired*. Several different devices may be used for acquiring these input clouds. The variety of types of sensors is the very first reason of the heterogeneity of point clouds.

We have already mentioned the *Microsoft Kinect*, which is a fixed sensor used to scan a scene through infrared structured lighting, recording depth in addition to color. The scans of such a device are generally of mediocre quality, tampered by many different defects, but acquired in real-time.

Optical laser-based range scanners and LiDAR scanners are more common devices.

They for example rely on the bouncing of a ray of light shot on the surface. Usually, the scanner automatically rotates and sweeps the surface. Complex scenes may be acquired by fusing several scans (registration markers are a way to make this process easier).

Figure 1.3: **Point cloud acquired by photogrammetry.** About 30 photos of the *owl* statue from the *EPFL Repository* [2] have been taken to produce the point cloud shown on the right (displayed in *MeshLab*).

Another interesting approach is photogrammetry. A standard camera is used to take usual matricial pictures of an object from several points of view. Algorithms can then be applied to the set of photographs in order to recover the 3D positions of corresponding sets of pixels in space. Figure 1.3 is an example of such an acquisition. This method is popular as it does not require any specific device apart from a common camera, the main drawback being the unusual defects produced, such as structured outliers.

1.2.2 Measurements

Depending on the type of sensor used, some additional information can also be acquired. These measurements are useful for retrieving the surface information on the neighborhood of data point.

The most common measurement is *normal vectors*: a normal vector is an estimation of the supporting plane of the infinitesimal surface beneath the point. It may or may not be oriented, depending on whether the relative position of the surface (behind or before the point) is known. Although many reconstruction techniques require normal vectors, not all scanners can provide reliable information on them. Therefore, normal vectors are often estimated through local analysis of the point cloud. Unfortunately, normal estimation is a

problem almost as hard as reconstruction itself, especially in the case of oriented normal vectors.

A sensor can also give a measure of *visibility*. If a sensor is based on ray shooting, for example, an acquired point was by definition visible from the sensor. Thus, the segment between a point and the sensor position is known to be outside of the surface, which can be a very useful hint for reconstruction.

Some material-aware sensors give measurements about *confidence*: according to the reflectivity of an object, for example, an estimated potential error of measurement may be computed. A statistical estimator may therefore adapt to the local confidence of the considered points to perform a reconstruction with variable precision.

Finally, *color* measurement can enrich the point cloud. Photogrammetry typically handles this: pixels are given a 3D position but still provide a color information. At first sight, this seems to be only useful for texturing and not reconstruction, but local color consistency may help segmenting the point cloud or consolidate the confidence of a reconstructed shape.

1.2.3 Defects

We saw that the input of reconstruction are point clouds that can be acquired by a large variety of sensors and that may embed different kinds of additional measurements. This diversity of input types results also in a variety of potential defects that are clarified in this section.

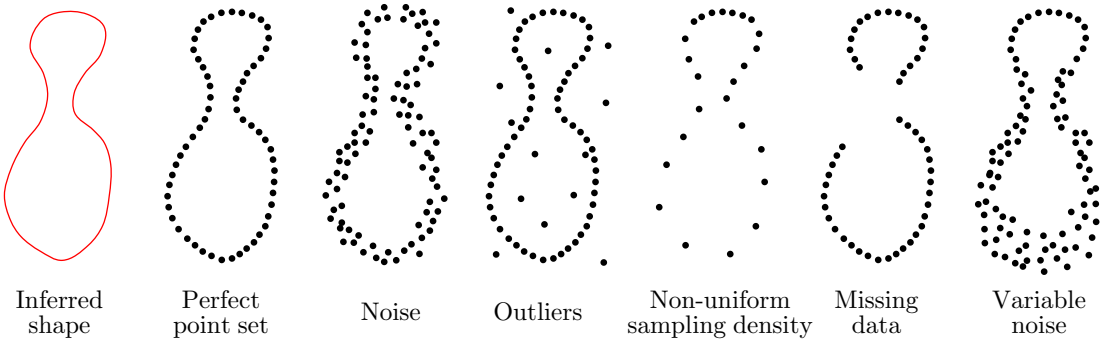


Figure 1.4: **Different types of defects of a point cloud.**

Noise

Because all sensors have physical limitations, they cannot provide perfectly accurate acquisition: as a consequence, all acquired point clouds are noisy, although this noise may be close to imperceptible for high-precision sensors.

The points are thus not located exactly on the inferred shape but on a neighborhood of it. The width of this neighborhood is called the *level of noise* and strongly relates to the scale of reconstruction as we shall discuss later.

Notable levels of noise can arise from registration of multiple scans or from a low-precision scanner or perturbations during acquisition (for example, moving).

Outliers

The term *outliers* comes from statistical analysis, as opposed to *inliers*. Outliers are points that do not relate to any part of the shape. They differ from noise in the sense that they can appear anywhere in space, potentially far away from the inferred shape, as can be seen on Figure 1.5.

Outliers can be produces by parasite shapes during the acquisition process (such as rain or dust) or false detection in photogrammetry. Figure 1.6 is an example of a point cloud with structured outliers produced by photogrammetry artifacts.

Non-Uniform Sampling Density

The density of points may not be constant throughout the acquired scene: according to many parameters, two objects of the same size might be sampled with a distinct number of points.

For instance, if the acquisition is performed from a single fixed sensor, surfaces facing perpendicularly the sensor are densely sampled whereas surfaces with a very low angle to the sensor are sampled with very few points.

As shown on Figure 1.7, density can also decrease with the distance to the sensor.

Missing Data

This defect can be seen as an extreme case of non-uniform sampling density: density might drop to zero in some parts of a scene. Therefore, parts of the inferred shape may not be represented at all by a subset of the point cloud.

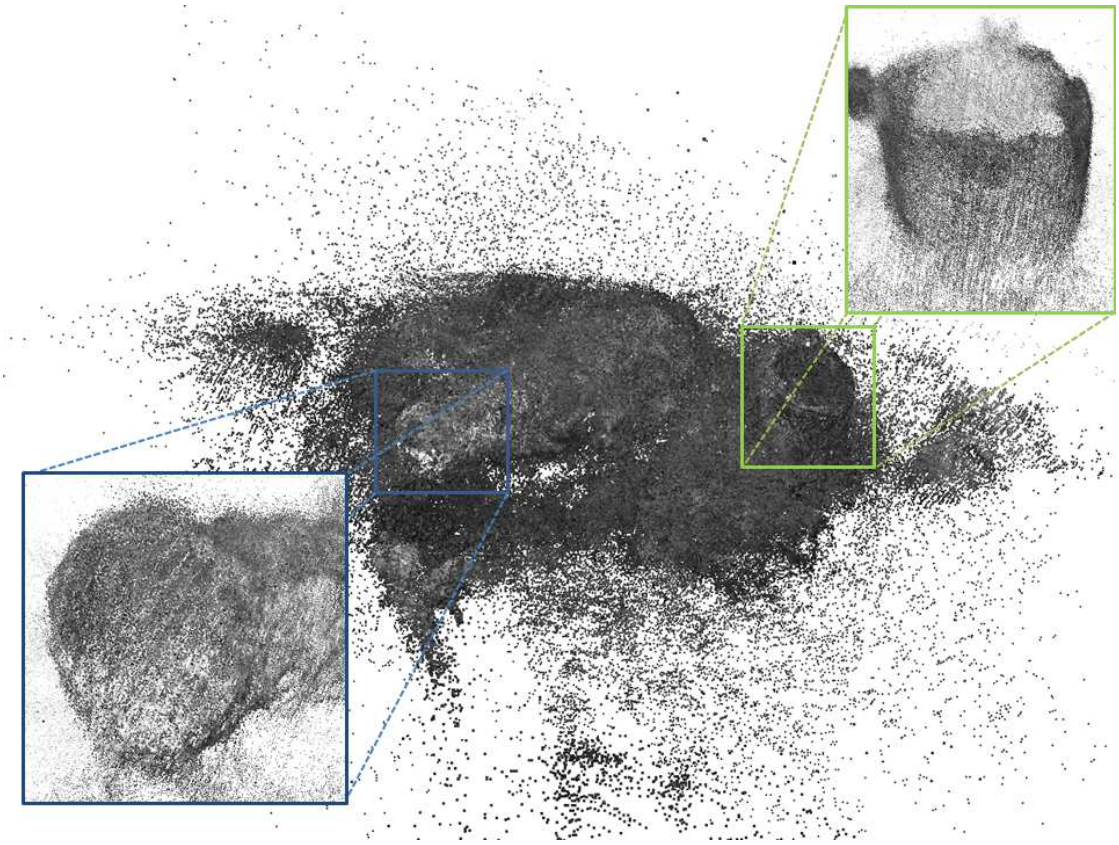


Figure 1.5: **Underwater point cloud ([39, 19]).** Acquired on a sunk ship, ridden with high noise and outliers.

This is the result of incomplete scans, happening for example with statues on the ground (Figure 1.8): the bottom of the statue can never be acquired by a vision-based sensor.

Variable Noise

In our study, we deliberately separate *variable noise* from regular noise as it raises very different problems: constant noise has been handled for a long time in the field of shape reconstruction through global approaches. Variable noise, on the contrary, requires local estimations of scale and the ability of adapting a fitting method to local levels of noise.

As we already saw, the *Kinect* device produces this kind of defect.

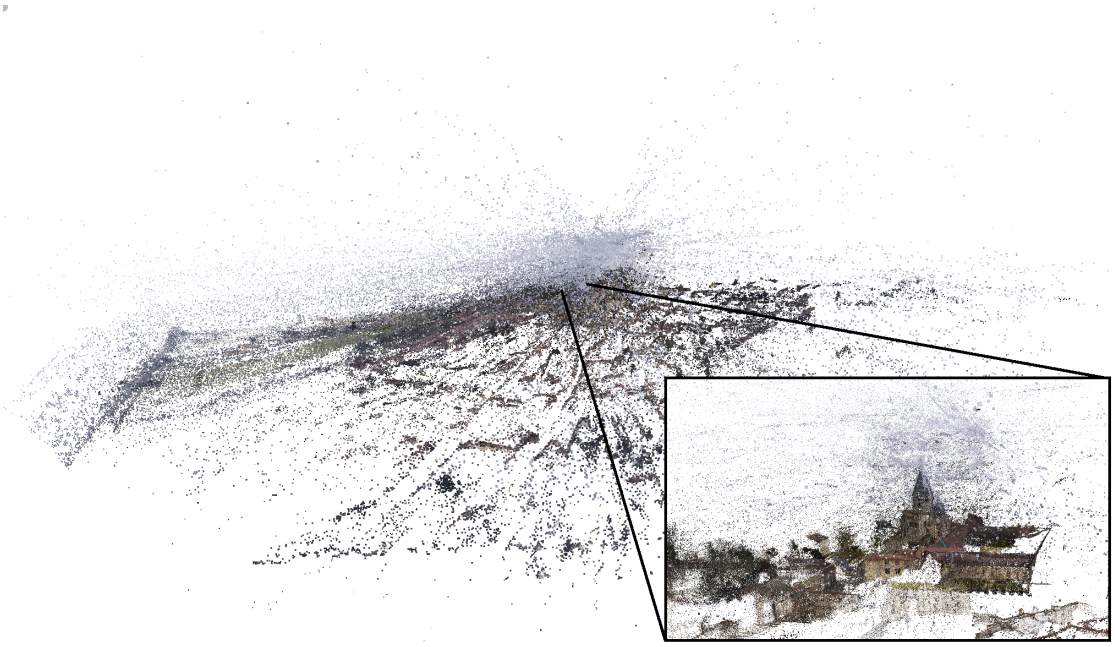


Figure 1.6: **Cluny Abbey with structured outliers.**

1.3 Output

The output of a shape reconstruction algorithm can take several different forms. They are classified here in three general denominations: explicit, implicit and semantic.

1.3.1 Explicit

The most natural way of representing a shape is an explicit formulation. *Splats* is a first approach to describe a reconstructed surface in 3D: atomic surface elements are grown around points. An example is given on Figure 1.9.

In the case of curve in 2D, a reconstruction may simply be a set of points, the vertices, and a set of segment, the edges, joining pairs of those points.

The equivalent for surface reconstruction is a 3D *mesh*. The principle remains the same while adding the third dimension: a mesh is a collection of vertices, edges and facets with

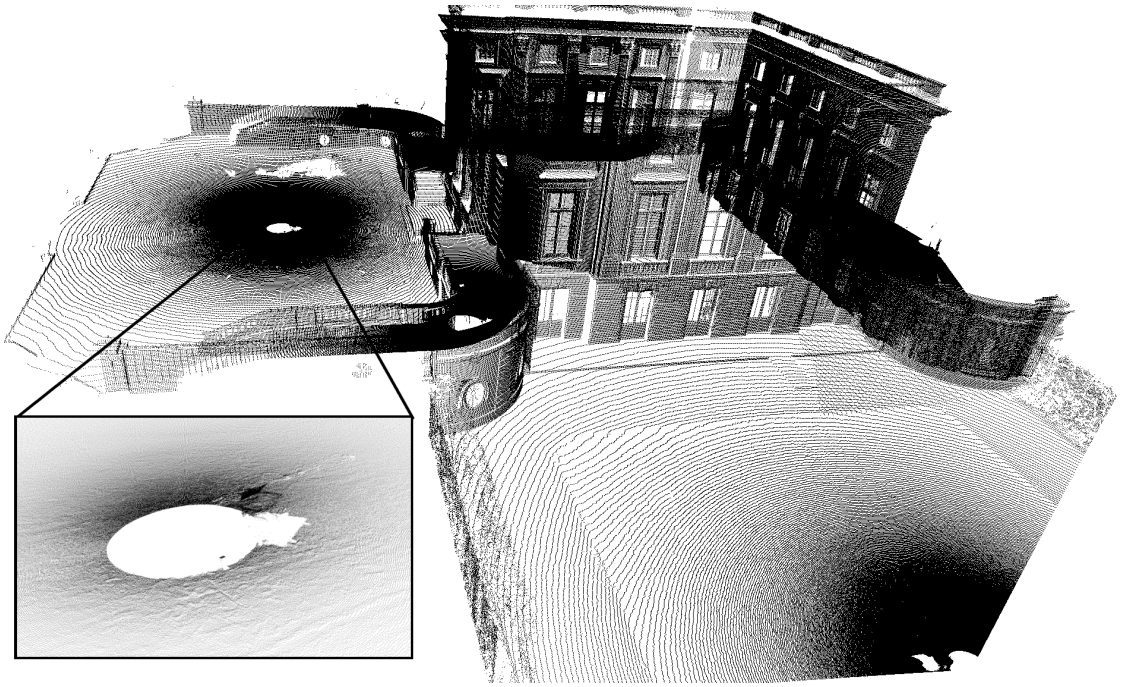


Figure 1.7: **Trianon point cloud.** Sensors were in the center of the black circles. Density is very high close to the center of those circles.

connectivity relationships. Facets are 2D polygons embedded in 3D space: triangles and quadrangles are the most commonly used.

The case of triangle mesh is an interestingly simple explicit formulation of a reconstructed shape. It is easily described through an indexed format and uses only simplices of each dimension. It takes advantage of a wide literature on the subject and relates to simple geometric objects: for instance, it can be a subset of a 3D Delaunay tetrahedralization.

If the triangle mesh does not contain any self-intersection, it correspond to a *simplicial complex* structure. Allowing for isolated simplices, this last explicit formulation allows for reconstruction of shapes with heterogeneous dimensions (3D scenes with surfaces, edges and isolated points, for example).

Figure 1.8: **Owl point cloud with missing data on the bottom** (courtesy *EPFL Computer Graphics and Geometry Laboratory* [2])

1.3.2 Implicit

Although less intuitive, the implicit formulation has many advantages over explicit ones. It separates the reconstruction computation from the representation of the output shape in itself and can be post-processed with more flexibility than meshes.

An implicit formulation is a description of where the shape is or where it is not. It can take the form an indicator function defined on the whole space. In this case, the shape is located on a isolevel of the function (see Figure 1.10).

In the case of *point set surfaces*, an operator of projection is defined according to the input point cloud. The surface in this case is implicitly defined as the stationary points of the projection operator.

Explicit representations of the surface are derived from these, using meshing algorithms

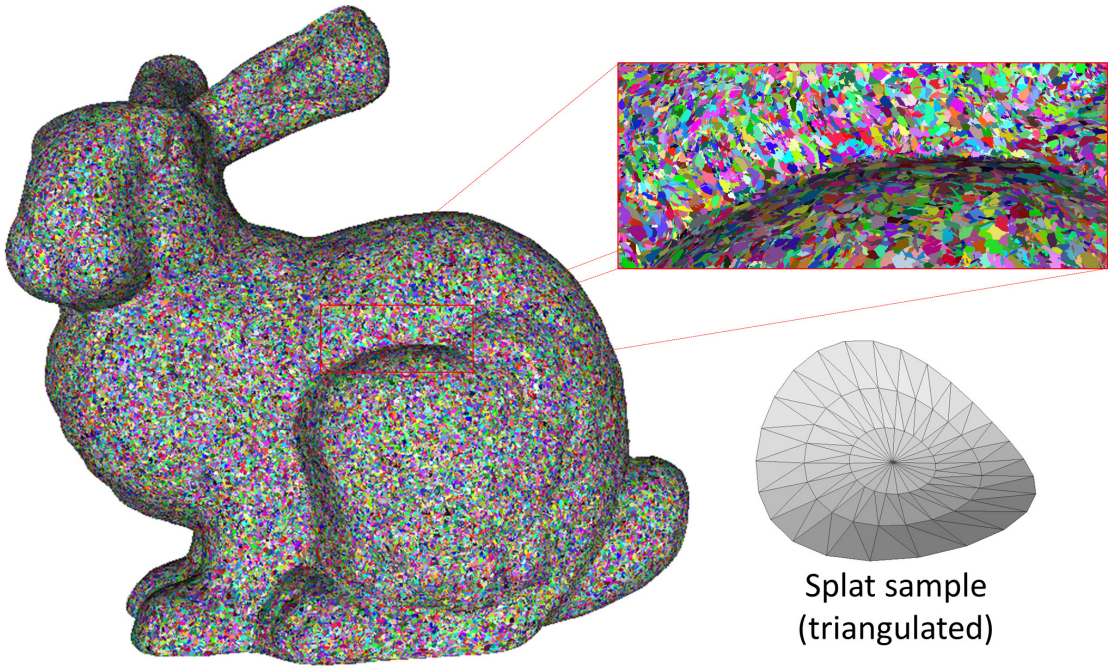


Figure 1.9: **Splat representation of the Stanford Bunny point cloud.**

such as *marching cubes* [60] that takes the implicit formulation as an oracle. An interesting point here is that the explicit representation is not unique: it depends on the meshing algorithm. This allows for a fine control of the output shape in terms of precision or iso- and anisotropy.

1.3.3 Semantic

Lastly, a shape can be described by a semantic formulation. This is an even higher level of abstraction than implicit surfaces.

Some methods recover collections of shape primitives. Identifying a simple shape sampled by a subset of the input is a *shape detection* problem. Reconstruction can be a post-processed version of a shape detection algorithm, adding connectivity and spatial coherence.

Using a predefined collection of complex shapes allows to simplify drastically the space of solutions: fitting segmented parts of the scene to a set of complex objects (chairs, tables, etc.) with some degrees of freedom in deformation. Although this can bring a high

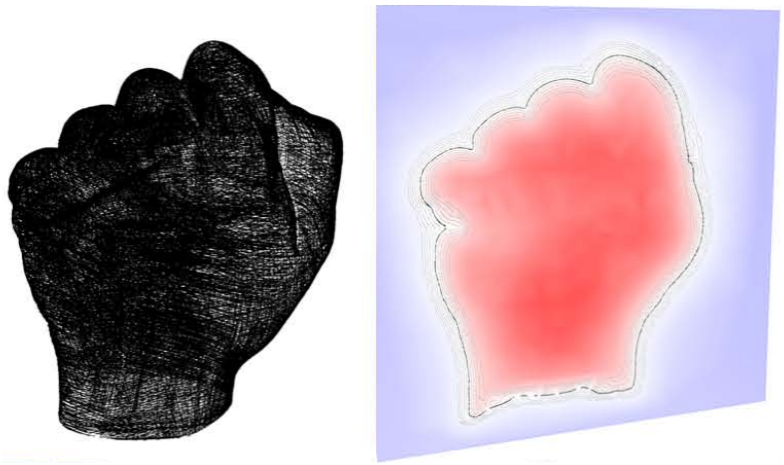


Figure 1.10: **Point cloud and implicit function on a slice.** The surface is defined by the black isovalue.

robustness to highly altered point clouds, it also narrows the types of input that can be handled.

This field of research also relates to *shape recognition* and with the problem of reliably labeling a complex scene with simple object descriptors.

1.4 Scientific Challenges

1.4.1 Complexity

With the increase of storage memory and of the broadband of sensors, the complexity of input point clouds has boomed in the last decade. Although reconstruction methods from the 90's could focus on input sets of a few thousands of points, we now have to take into account very large scans. Entire parts of cities can be digitized using up to billions of points.

If the available storage memory is still regularly increasing, the speed of CPU has reached a bound. This lead to a subtle transformation of the general behavior of algorithms nowadays: the main need becomes time efficiency, even at the cost of storing lots of additional variables or preprocessed information.

This context of high input complexity calls for methods with a low time dependence

on the size of the input. Emphasis is put on multiscale and parallel approaches.

1.4.2 Details

The complexity of the input point clouds is often the expression of a high level of details. Not only a scene can be described with several billions of points, but it can embed very small features of a few dozens of points only, for example.

While it seems easy to handle simple figure regardless of the number of points (a plane remains a simple plane whether it is defined by 3 points or by 3 billions of points), the extent of the range of possible scales on a single scene remains one of the main challenges in shape reconstruction.

More specifically, there is a huge gap of difficulty between reconstructing a building only defined by its walls and another containing windows and balconies.

1.4.3 Robustness

The last challenge which seems interdependant from the previous one is the presence of a large variety of defects, as described in Section 1.2.3: the problem of reconstructing details reliably is even more of a challenge when the local level of noise compares to the scale of details.

Their exists many very specific methods focusing on point clouds with particular defects, as we discuss in Section 2.3. Unfortunately, they often take advantage of additional measurement and very few can handle a point cloud with little assumption on its properties.

Although it seems impossible to handle every possible defects at once, their is still a need for a general framework to achieve robustness with as little *a prioris* as possible on the input point cloud.

1.5 Focus

As we discussed on this section, shape reconstruction is very wide and diversified topic. Consequently, this thesis is only focused on a specific part of the problem.

Our main focus is robustness, using as little assumptions as possible. First, reconstruction is taken on its most general meaning: we do not focus on specific point clouds such as urban or indoor scenes. Secondly, we only require raw point clouds with no additional measurements (no normal vector or visibility information).

Furthermore, we do not have any requirement about the defects of the point clouds: it can be ridden with noise, outliers, missing data, etc. The only defect we choose to ignore is widely variable sampling density, as it does not seem possible to handle this altogether with outliers.

We reconstruct both implicit (Part I) and explicit (Part II) shapes. Part I is about smooth closed shapes whereas Part II describes a method to handle piecewise-planar shapes with boundaries.

1.6 Publications

This thesis has led to the publication of the following articles:

- [37] Simon Giraudot, David Cohen-Steiner, and Pierre Alliez. Noise-adaptive shape reconstruction from raw point sets. In *Computer Graphics Forum*, volume 32, pages 229–238. Wiley Online Library, 2013
- [4] Pierre Alliez, Simon Giraudot, and David Cohen-Steiner. Robust shape reconstruction and optimal transportation. In *Courbure discrète: théorie et applications*, volume 3, 2014
- [*Submitted*] Simon Giraudot, Mathieu Desbrun, David Cohen-Steiner, and Pierre Alliez. Piecewise-Planar Surface Reconstruction via Convex Decomposition. 2015

Note that Chapters 9.2 presents a transitional work that was not published as it did not provide satisfactory results. However, we include it in this document for its scientific interest and for the intuition it gave for the following developments.

I

Noise-Adaptive Reconstruction

2

State of the Art

2.1 Main Approaches

There is a vast literature on smooth closed shape reconstruction from unorganized point clouds. In order to get a meaningful overview, we propose next a taxonomy which distinguishes between the main approaches for tackling the reconstruction problem.

Interpolant

Going back to the initial research on shape reconstruction, the first approaches were mostly *interpolant*: in this view, reconstruction is formulated as the problem of "connecting the dots". These approaches, pioneered by researchers in Computational Geometry, produce output shapes based on common geometric data structures. The Delaunay triangulation is a relevant example as it is related to the Voronoi diagram which encodes neighboring points. Hence, if the sampling is dense enough and with the assumption that the shape is "simple", then a subset of the Delaunay simplices provides a plausible reconstruction.

The main advantage of such methods is to provide proven guarantees to recover the correct topology and geometry. However, these guarantees rely upon ideal assumptions such as dense sampling and absence of noise, that are rarely met in practice. Point clouds

acquired from measurements are often sampled with variable density and are hampered with a wide range of defects that cannot be dealt with by these interpolant methods. If the input point cloud is hampered by uncertainty, there is no reason to interpolate the points anymore.

Variational

Robustness to variable sampling and noise has motivated another line of approaches, referred to as *variational*. These approaches take advantage of global solves, the output surface passing near the input points instead of interpolating them. The problem of "connecting the dots" is not relevant anymore: the input points are only considered as witnesses that provide information on the shape location with a certain level of confidence.

The main idea behind these approaches is to search, in the space of functions defined from $\mathbb{R}^n$ to $\mathbb{R}$, a so-called indicator function. This function usually applies to closed shapes and distinguishes the inside from the outside of the shape. The output shape is often defined by contouring an isovalue of the function, which is possible if this function is well-behaved with a non-zero gradient.

One interesting property is that when such a function is defined everywhere, even away from the shape: it provides a suitable input for meshing and contouring algorithms. The smoothness of the function is also suited to match the common assumption that the inferred shape is smooth.

This usually provides a fair level of robustness to noise and sometimes to outliers. As many of these approaches further constrain the output to be a closed smooth shape, they are to some extend also robust to missing data: by construction, they remain hole-free and also intersection-free.

Point Set Surfaces

Shapes may also be reconstructed through so-called *point set surfaces*. As for variational approaches, the input points are seen as witnesses, but the computed function differs: point set surfaces are defined as the fixed point of a projection operator.

Intuitively, it can be seen as a search for a stable function close to the input points. For this reason, the function of a usual point set surface is only defined on the close neighborhood of the inferred shape, although some methods extended this definition to the whole domain.

Note that the projection operator is defined by the input data, whereas variational approaches define arbitrary functions where input points are only used to estimate the functions.

This offers a finer control over the role played by the input points: point set surfaces have been initially devised to deal with noise and non-uniform sampling.

Volumetric

Some variational approaches differ slightly from the ones we already presented, and can be classifies as *volumetric*. The inferred shape is assumed to bound a solid object (a surface in 2D or a volume in 3D, hence the term "volumetric").

This assumption yields further refinement of the surrounded volume (or surface). For example, volumetric consistency can be taken into account to estimate the probability that two surface sheets surround the same volumetric part. This provides additional knowledge on the adjacency relationships among the reconstructed surface.

These approaches often outperform on input point clouds where parts of the data is missing. Nearby surface sheets that are difficult to separate in a variational sense can be better handled.

Primitives Fitting

As mentioned in the introduction, an input scene can be considered as made up of canonical primitives with specific relationships: structural such as adjacency, or geometric such as regularities. In some cases, reconstruction is a post-processing of a shape detection algorithm. A family of reconstruction methods focus on this *primitive fitting* point of view.

In this view, the input point cloud is studied at a more global level. The non-local regularity yields a higher resilience to defects such as missing data, non-uniform sampling density or noise.

This however comes at the cost of strong constraints for the output and of a high dependency on the quality of the shape detection.

Complex Object Fitting

One step forward to a semantic assumption is the use of dictionaries of precomputed complex shapes to perform reconstruction: man-made shapes, furnitures, architectural

parts, etc.

This family of methods usually provides a high robustness to many defects but the assumptions on the input point clouds are very domain-specific. For example, indoor scenes are well reconstructed using a dictionary of usual apartment furnitures.

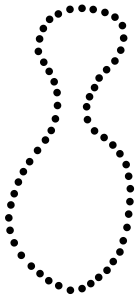
Although these algorithms have many applications to high-level semantic algorithms such as shape recognition and labeling, their narrow scope only partially relates to some extend to general reconstruction, and touches the topic of object recognition.

Focus

In our review of the state-of-the-art, we focus on reconstruction of smooth closed shapes from point clouds. Instead of targeting completeness, we describe the main rationale behind the various approaches devised to handle specific properties of point clouds.

We first review the methods designed for perfect point clouds, then review the different approaches devised to deal with various types of defects. Note that we provide another review on Part II, focused on piecewise-planar reconstruction.

2.2 Perfect Data



With the increasing range and amount of defects in input point clouds, algorithms designed to reconstruct from defect-free input point clouds have become less satisfactory. Nevertheless, their scientific value is high as they often provide mathematically proven guarantees.

These methods are said *interpolant*, which means that the output surface passes through the input points.

Crust

The crust algorithm was proposed by Amenta et al. [6] for curve reconstruction in 2D.

It relies on the observation that a subset of the edges of $\mathrm{Del}(\mathcal{P})$ is a valid reconstruction and that the vertices of the Voronoi diagram $\mathrm{Vor}(\mathcal{P})$ can be seen as a discrete approximation of the continuous medial axis of the inferred shape.

Because they are located close to the medial axis, adding the vertices of $\mathrm{Vor}(\mathcal{P})$ to $\mathrm{Del}(\mathcal{P})$ removes the edges of this triangulation that are not part of the reconstruction.

The pseudo-code of the algorithm reads (see also Figure 2.1):

1. Compute the Voronoi diagram of the input point cloud $\mathcal{P}$.
2. Compute the Delaunay triangulation of the combination of the point cloud and of the Voronoi vertices.
3. The edges of this triangulation connecting two input points are the reconstructed curve.

Beyond the noise-free assumption, this algorithm is valid and proven only under an assumption on sampling density. In this case, it consists in an upper-bound on the distance between two input points. One remarkable property is that this upper-bound is a function of the *local feature size*, that captures curvature, thickness and separation altogether. The inferred shape does not need to be sampled densely everywhere: it only needs to be densely sampled on areas with a high local shape complexity (with small local feature size) whereas fewer sample points are required in areas with large local feature size.

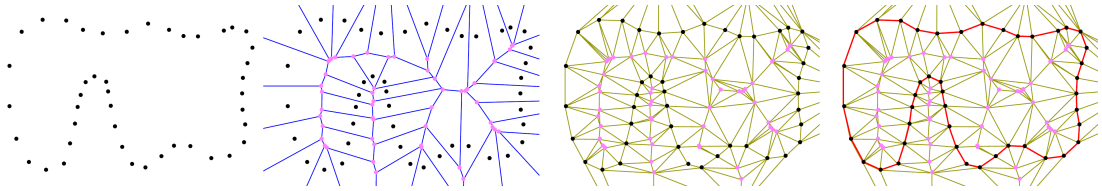


Figure 2.1: **Crust algorithm [6].** From left to right: input point cloud; Voronoi diagram; Delaunay triangulation of input points + Voronoi vertices; edges between two input points (in red) form the reconstructed curve.

This algorithm was later extended to the reconstruction of surfaces in 3D [5]. However, it does not generalize directly due to a different behavior of $\mathrm{Vor}(\mathcal{P})$ in 3D: some Voronoi vertices might be located too close to the surface and "break" edges and facets of $\mathrm{Del}(\mathcal{P})$ that should belong to the reconstruction. This problem is answered by observing that a subset of the vertices of $\mathrm{Vor}(\mathcal{P})$ provide an approximation of the medial axis. An additional filtering step is therefore required before applying an algorithm similar to the one in 2D.

From a more general point of view, algorithms in 2D rarely generalize in 3D without further processing. Attali [9] propose a shape reconstruction using a subset of the Delaunay graph with a proven exact solution in 2D. However, 3D reconstruction requires additional heuristics, such as triangulating the borders of the Delaunay graph subset or inferring the volume enclosed by the inferred surface.

Ball pivoting

The *ball pivoting* method was proposed by Bernardini et al. [13] for surface reconstruction.

Its advantage is a linear time complexity, making it suited to large input point clouds. The main idea is to simulate a ball of a certain radius ρ pivoting around the input points and generating an output triangle every time three points are simultaneously touched by the ball.

In practice, this strongly relates to α-shapes [34]: an α-shape is a subset of $\mathrm{Del}(\mathcal{P})$ where the simplices have a circumscribing sphere of radius $r \leq \alpha$. More intuitively, considering the space as "filled with matter", the α-ball can be seen as an eraser that cannot cross a point of $\mathcal{P}$. The only remaining parts are thus around points that are closer to each other than α, giving an approximation of the shape for a properly chosen value of α.

The ball pivoting algorithm relies upon the same principle with the major advantage of not having to compute Del($\mathcal{P}$):

1. A ρ-ball is initialized on a seed triangle.
2. It pivots around two of the triangle points until another point is hit, producing a new triangle (Figure 2.2).
3. This is repeated until every pivot has been tested, producing an output 3D triangle mesh.

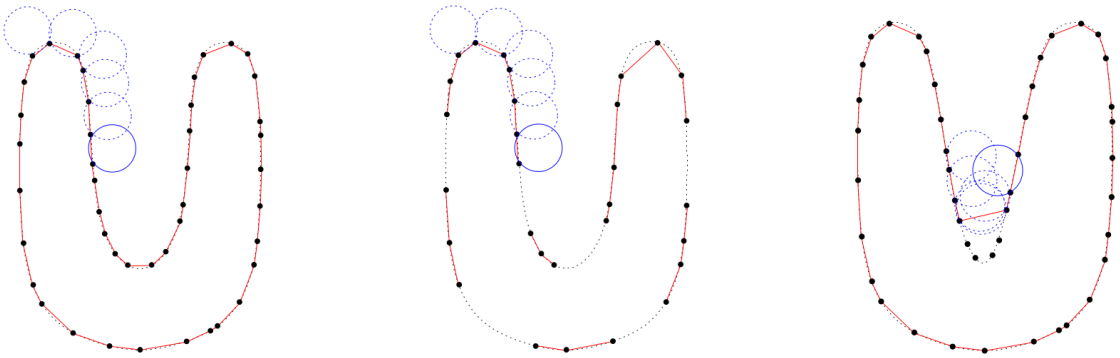


Figure 2.2: **Ball pivoting algorithm [13].** From left to right: algorithm on a perfect point cloud; point cloud with lower sampling density; point cloud sampled on a shape with higher curvature.

The possibility for the user to adjust a scale parameter definition is one improvement compared to the Crust algorithm. It means that the radius of the ball ρ can be adjusted to handle different densities of sampling and, to a small extend, noise.

Overview

Interpolating methods often rely on data structures that are common in computational geometry. Using the properties of simplicial complexes based on the input point cloud, guarantees are proven on the validity of the output reconstructions.

The initial approaches use constructions that are uniquely defined for a given point cloud, such as Delaunay triangulations or Voronoi diagrams. Addressing the absence of

user-control on the output seen as major limitation, some methods derived parameter-dependent methods, such as α-shape and ρ-ball pivoting. These approaches pioneered the use of a user-specified scale in order to handle different types of input with low noise.

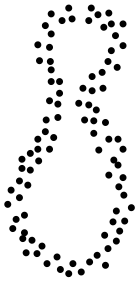
Note that some interpolant methods are robust to noise. They are based on a preprocessing of the input and will be treated in Section 2.3.1. However, as these methods remain interpolant, robustness to defects cannot be fully achieved as an uncertain input *de facto* produces an output hampered by a similar uncertainty. This is the main motivation for devising methods that are non-interpolant and that we present next.

2.3 Dealing with Defects

In this thesis the heart of the problem is to deal with imperfect point clouds. Robustness to defect-laden point clouds has been a topic of increasing interest in recent years.

We propose a taxonomy centered on the the main type of defects dealt with by the methods described. Our terminology in terms of defects matches the one provided in Section 1.2.3.

2.3.1 Noise



Because acquisition devices can only provide finite precision, noise is unavoidable when dealing with acquired point clouds. This is why most reconstruction methods can at least handle moderate levels of noise. We review next the main reconstruction approaches together with some intuition on how the noise is handled.

Point Set Surfaces

The notion of *point set surfaces* (PSS) is by itself a topic in shape reconstruction. It was first defined by Alexa et al. [3] who introduced an operator that project points on a approximated *moving least squares* (MLS) surface.

The main idea behind MLS can be summarized as follows: a local fit is computed at each query point in a local frame; then, a function of the distance to each query point is defined to weight its contribution; the local fits are finally put together by minimizing an energy measuring the approximation error.

The MLS projection proceeds as follows:

1. **Reference domain:** a local reference domain is computed (a plane for now). The local plane $H = \{\boldsymbol{x} | \langle \boldsymbol{n}, \boldsymbol{x} \rangle - D = 0, \boldsymbol{x} \in \mathbb{R}^3\}, \boldsymbol{n} \in \mathbb{R}^3, \|\boldsymbol{n}\| = 1$ minimizes a local weighted sum of square distances from the points of $\mathcal{P}$ to the plane. At query point $\boldsymbol{q}$, H minimizes the following expression:

$$\sum_{i=1}^{N}(\langle \boldsymbol{n}, \boldsymbol{p_i}\rangle - D)^2\theta(\|\boldsymbol{p_i} - \boldsymbol{q_H}\|),$$

where $\boldsymbol{q_H}$ denotes the projection of $\boldsymbol{q}$ on H and θ is a smooth, radial, monotone decreasing function, positive on the whole space. An approximation of the tangent plane to S near the point $\boldsymbol{q}$ is used as the local reference domain at point $\boldsymbol{q}$.

2. **Local map:** a local bivariate polynomial approximation of the surface in a neighborhood of $\boldsymbol{q}$ is computed on this reference domain. The polynomial approximation g minimizes the following weighted least squares error:

$$\sum_{i=1}^{n}(g(x_i, y_i) - f_i)^2\theta(\|\boldsymbol{p_i} - \boldsymbol{q}\|),$$

where (x_i, y_i) denotes the coordinates of the projection of $\boldsymbol{p_i}$ in the local coordinate system of H, and f_i denotes the distance of $\boldsymbol{p_i}$ from H.

Levin [56] proved that the surface defined by the stationary points of the projection operator is a two-dimensional manifold and that the resulting surface is infinitely smooth if $\theta \in C^\infty$.

The main limitation is that the projection operator is only valid near the surface and that a reliable initial guess of the surface location is needed. In the case of a noisy point cloud, using a random input point may not provide reliable result as a point can lie at an arbitrary large distance from the inferred surface.

Amenta and Kil [7] studied the domain of a point set surface to provide a reliable starting point. They showed that a Gaussian function for θ provides robustness to noise: the intuition is to grow the size of the Gaussian function until the noise level is captured. Figure 2.3 shows the quality of plane-fitting for a noisy point cloud using this method.

Many variants of PSS have been developed such as *progressive PSS* for increasing precision on noise-free parts with small details [36]. Other variants are presented in the following chapters, related to other defects (Section 2.3.3).

Poisson Reconstruction

Variational approaches are another very prolific field of research. Among them, the Poisson Reconstruction [48] is popular, and has been extended and improved over the

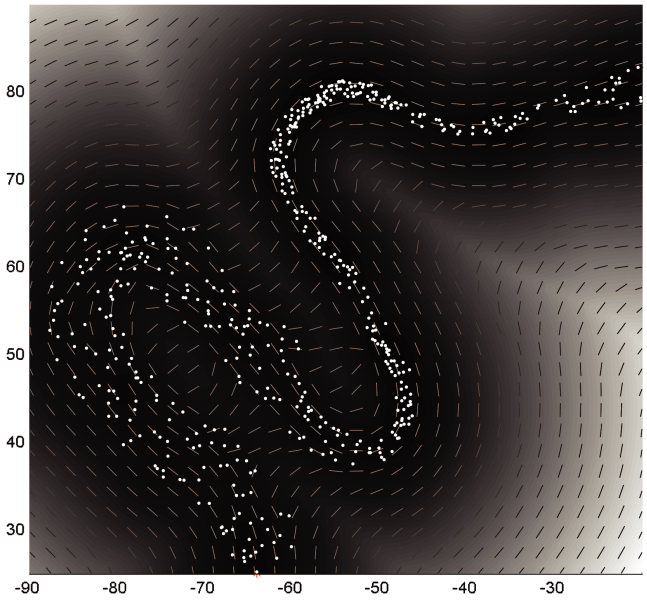


Figure 2.3: **The domain of a point set surface [7].** The background intensity depicts the MLS energy function defined on the domain. The gray lines are the local best fitting lines H found.

years.

The Poisson reconstruction consists in computing an indicator function f whose gradient accurately fits the normal vector field $\boldsymbol{n}$ of the oriented point cloud, i.e., $\min_f \|\nabla f - \boldsymbol{n}\|$.

Applying the divergence operator, this boils down to a Poisson problem: $\Delta f = \nabla \boldsymbol{n}$. The indicator function f is piecewise constant with opposite signs inside and outside the inferred shape, its gradient is not well defined on the shape. The goal is thus to compute an approximated smoothed version of the ideal indicator function, using a smoothing filter θ:

$$\Delta(f * \theta)(q) = \int_{\partial M} \theta(\boldsymbol{q}) \boldsymbol{n}_{\partial M}(\boldsymbol{p}) \mathrm{d}\boldsymbol{p},$$

where ∂M denotes the boundary of the solid M.

Note that using this filter θ favors the smoothness of the estimated indicator function f.

To compute the indicator function, the space is discretized using an octree with a user-specified maximum depth: varying this parameter provides the user with control on the level of details of the output reconstruction as shown by Figure 2.4.

Further improvements have been proposed. For example, a streaming variant can deal with massive data sets [14]. A GPU variant has also been devised to speed up the computations [83].

Finally, a recent improvement is the Screen Poisson Reconstruction [49]. The original Poisson reconstruction is robust to noise at the cost of oversmoothing. This is addressed by providing the user with soft control over a parameter to favor data fidelity. Introducing a new parameter λ that trades interpolation for approximation, the minimized energy becomes:

$$E(f) = \int \|\boldsymbol{n}(\boldsymbol{p}) - \nabla f(\boldsymbol{p})\|^2 \mathrm{d}\boldsymbol{p} + \lambda \sum_{\boldsymbol{p_i} \in \mathcal{P}} \|f(\boldsymbol{p_i}) - 0\|^2.$$

Introducing a two-terms energy may be seen as a new step forward in the quest for noise-adaptive approaches: a trade-off between smoothing and interpolating the input points.

Robust Cocone & Scale Space

A few interpolating methods have been devised to yield noise robustness. The main rationale is to make the interpolation a better-posed problem.

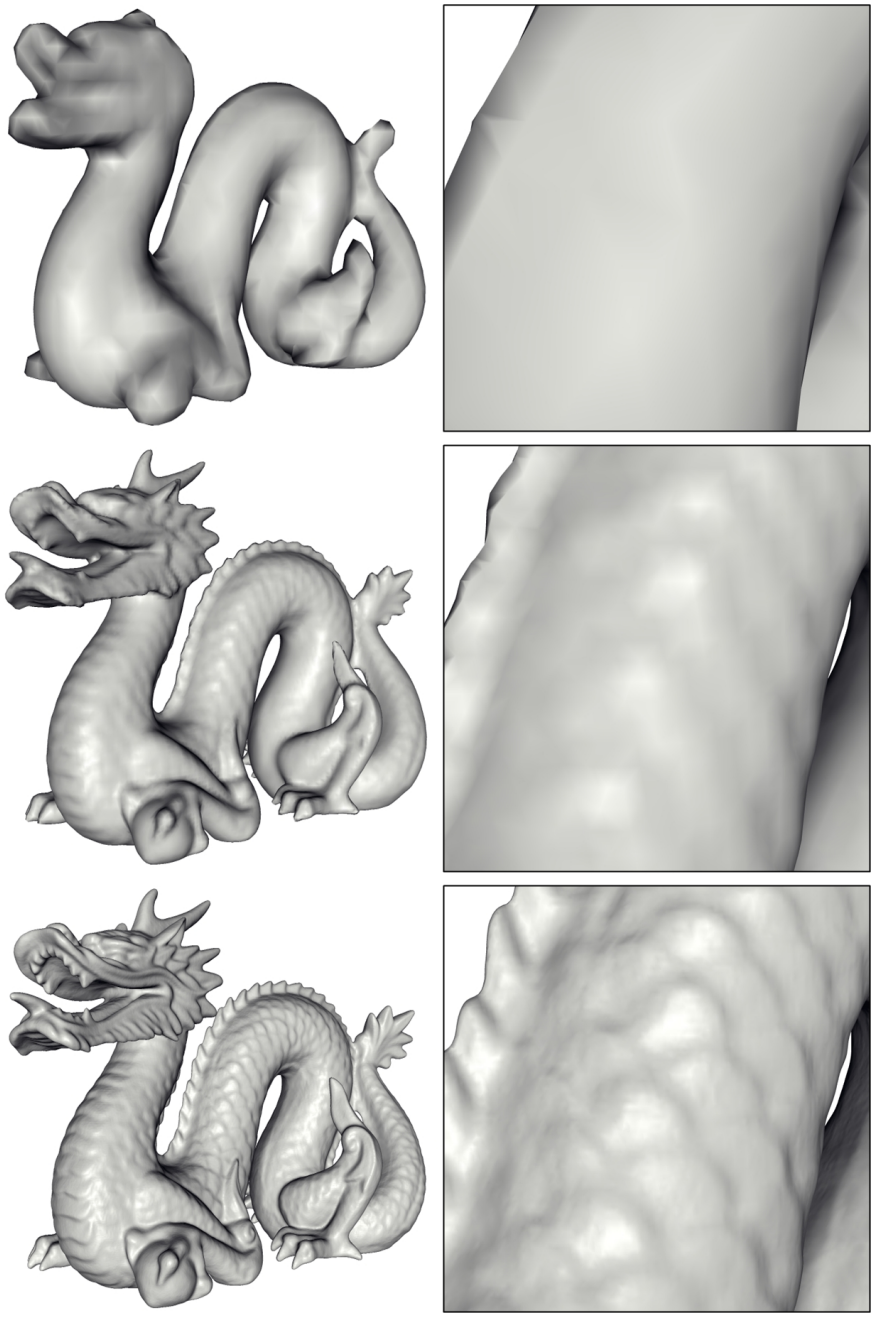


Figure 2.4: **Poisson reconstruction [48].** The proper level of noise can be captured by selecting the octree depth (from top to bottom, 6, 8 and 10).

Dey et al. [29] introduced the notion of "robust cocone". The properties of the Delaunay triangulation and more specifically of the Delaunay balls are used to distinguish which input points are in the internal part of the noise of the point cloud.

It is proven that, given a certain noise model, the union of large Delaunay balls is homeomorphic to the underlying surface (Figure 2.5 provides a visual depiction).

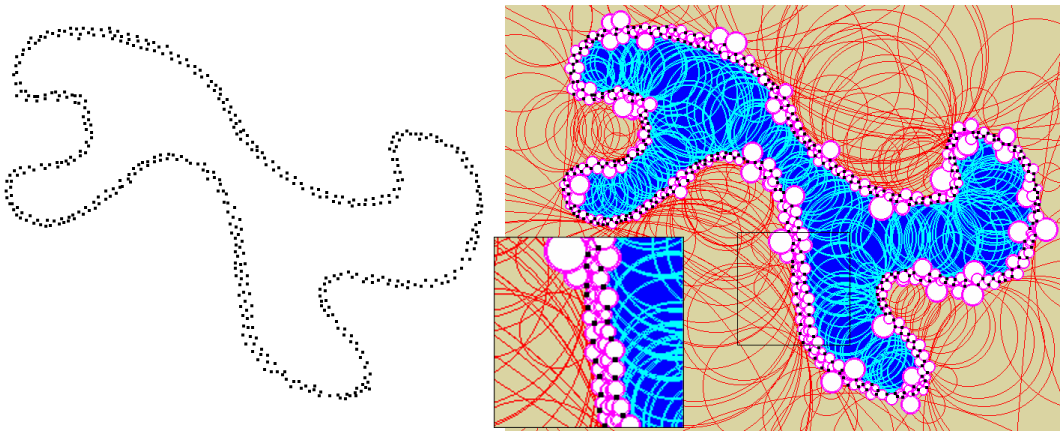


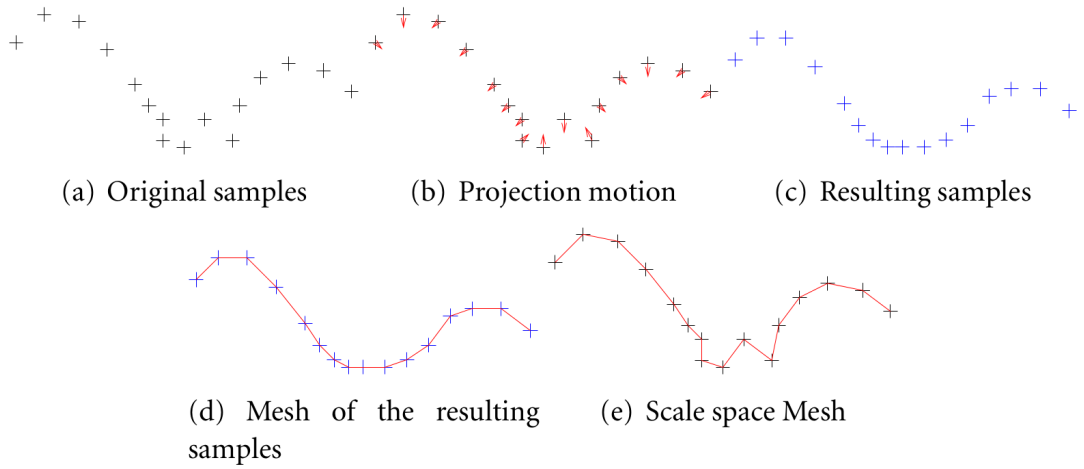
Figure 2.5: **Robust cocone [29].** On a noisy point cloud, points inside the noise band have small Delaunay balls compared to points on the border.

Digne [30] starts by preprocessing the input point cloud through the use of a scale space. First, the noise is removed through a series of smoothing operators to make the problem better-posed: the input points are connected via the ball pivoting algorithm on the smoothest version of the point cloud. The connectivity of this reconstruction is then propagated to the less smooth levels in order to recover small features until the desired level of details is reached. This is shown on Figure 2.6.

Overview

Dealing with noise is a requirement for most reconstruction methods. The initial idea is to select a scale parameter according to the level of noise. Note however that this parameter is often a variable of the reconstruction method and not specifically designed for handling noise.

Because the noise and the level of details of a shape are not always correlated, some methods have evolved to provide a finer control over the handling of noise to the user,



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

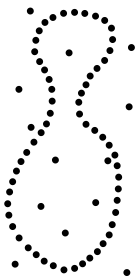
(d) Mesh of the resulting samples (e) Scale space Mesh

Figure 2.6: **Scale space [30].** The shape is correctly reconstructed by transiting through a smoothed version of the point cloud.

independently of the scale of details. This raises a question of noise robustness: how to reliably smooth the noise without altering the small details of a shape?

In previous work, dealing with noise often requires adjusting a parameter. A principled way to select these parameters is not always well defined and often require a trial-and-error process. Automatic parameter selection is an open problem. We discuss this specific issue along with the problem of variable noise in Section 2.3.5.

2.3.2 Outliers



Robustness to outliers is quite different from noise robustness as some points of the input must be discarded or ignored. Some methods aim at explicitly removing outliers through filtering or data clustering [77]. Other use statistical tools or metrics that are implicitly robust to outliers.

Local Outlier Factor

Outliers are an obvious non-relevant part of the input point cloud. A first rationale may be to detect and remove them explicitly.

Sotoodeh [78] set up an outlier detection algorithm based on density analysis [18]. It uses some distance functions known to be robust to outliers under some conditions:

- The K-distance: the distance from the K^{th} nearest neighbor to a query point $\boldsymbol{q}$;
- The K-distance neighborhood: the size of the minimal ball containing the K-nearest neighbors of a query point $\boldsymbol{q}$.
- The reachability distance between 2 points $\boldsymbol{a}$ and $\boldsymbol{b}$: the maximum value between the K-distance from b and the usual Euclidean distance, $\text{reach} - \text{dist}_K(\boldsymbol{a}, \boldsymbol{b}) = \max\{K - \text{distance}(\boldsymbol{b}), d(\boldsymbol{a}, \boldsymbol{b})\}$.

This last distance has the advantage of being more stable. The reachability distance from two far away objects is simply their actual Euclidean distance whereas objects in a K-distance neighborhood have a constant value that is $K - \text{distance}(\boldsymbol{b})$.

The statistical fluctuations of the Euclidean distance for all points close to the query is therefore significantly reduced, with a magnitude controlled by the parameter K.

The *local reachability density* at query point $\boldsymbol{q}$, which can be seen as the average *reachability* distance based on the K-nearest neighbors of $\boldsymbol{q}$, is defined this way:

$$lrd_K(\boldsymbol{q}) = 1/\left(\frac{\sum_{\boldsymbol{p}\in N_K(\boldsymbol{q})} \text{reach} - \text{dist}_K(\boldsymbol{q}, \boldsymbol{p})}{|N_K(\boldsymbol{q})|}\right).$$

Finally, the *local outlier factor* (LOF) is defined as:

$$LOF_K(\boldsymbol{q}) = \frac{\sum_{\boldsymbol{p}\in N_K(\boldsymbol{q})} \frac{lrd_K(\boldsymbol{p})}{lrd_K(\boldsymbol{q})}}{|N_K(\boldsymbol{q})|}.$$

This function of a point $\boldsymbol{q}$ is the average of the ratio between the local reachability density of $\boldsymbol{q}$ and those of its K-nearest neighbors. It captures the degree to which this point is called an outlier.

A point located in a high density area – supposedly where the inferred shape lies – has a value of LOF close to 1, whereas outliers are associated to larger values.

Computing this function for every input point and applying a threshold yields an explicit binary partition of the input between inliers and outliers (see Figure 2.7). Note that this method requires a user-defined parameter K relating to the outlier density. This parameter is adjusted through a trial-and-error process.

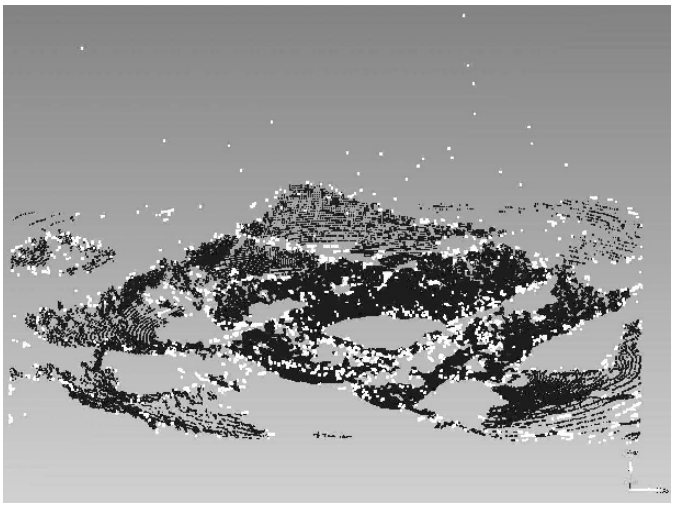


Figure 2.7: **Outliers filtered by LOF [78].** Detected outliers depicted in white, relevant data in black. $K = 10$ on this example.

Locally Optimal Projection

Lipman et al. [59] contributed the so-called *locally optimal projection* approach.

An arbitrary point set is projected onto the input point cloud: the goal is to find the set of projected points $\boldsymbol{q}$ that minimizes a sum of weighted distances to the points $\boldsymbol{p_i}$ of the input $\mathcal{P}$, with respect to radial weights centered at the points in $\boldsymbol{q}$. They are defined as the stationary points of the projection operator $Q = G(Q)$ where $G(C) = \arg\min_X E_1(X, \mathcal{P}, C) + E_2(X, C)$.

E_1 denotes an energy term that drives the points of $\boldsymbol{q}$ to approximate the geometry of $\mathcal{P}$ while E_2 favors a fair distribution of the points of $\boldsymbol{q}$ all over the inferred shape.

In order to discard outliers, the chosen function for E_1 is closely related to the multivariate median, which minimizes the sum of Euclidean distances to a data set S:

$$\boldsymbol{q} = \arg\min_X \{\sum_{i=1}^{n} \|\boldsymbol{s_i} - \boldsymbol{x}\|\}.$$

This median has proven robust to outliers, compared to the standard "mean" average (L^1 median). Figure 2.8 depicts a visual comparison of this method with the common MLS.

A point set Q is sought after to represent the geometry of the point cloud using this median on local subsets of the point cloud with a fast-decaying weight function θ: while

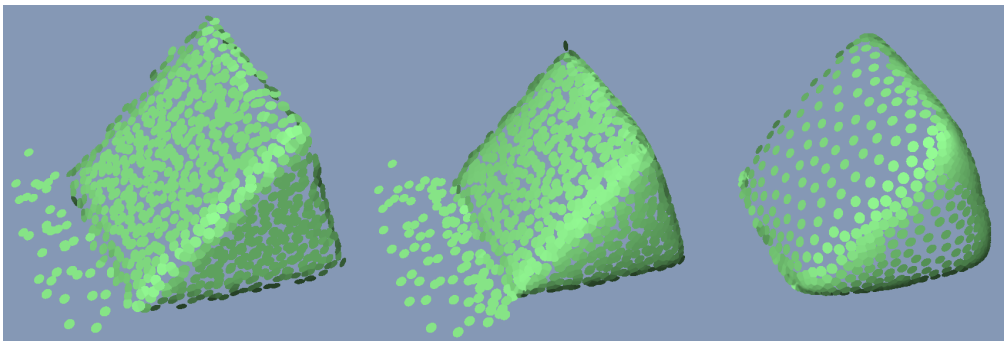


Figure 2.8: **Locally optimal projection [59].** Left: input point cloud with ghost-geometry noise on the side. Middle: MLS. Right: locally optimal projection.

averaging on a sufficiently large support allows for noise robustness, using the median operator allows for outlier robustness.

L^1-norms have been used in wide ranges of algorithms ([53, 50, 62]) for their ability to deal with noise and outliers. A feature-preserving variant is detailed in Section 6.4.

Signing the Unsigned

Mullen et al. [64] tackle outlier robustness through a robust distance function based on optimal transportation theory.

Instead of using a standard distance function to a point cloud, the robust variant computes the average of the square distances to the K nearest neighbors. This implicitly makes outliers irrelevant for well suited value for K. Figure 2.9 depicts a reconstruction from a point cloud hampered with structured outliers.

This approach is detailed in Chapter 4.

Overview

Handling outliers calls for specific metrics and analysis tools. Noise is commonly handled through smoothing and averaging on small neighborhood, but reconstruction of outlier-ridden data requires different tools to avoid generating reconstruction with many artifacts.

Averaging is not sufficient to deal with outliers, as far away points may lead to significant distortions. This motivates the use of robust norms such as the L^1-norm or robust distances such as the K-distance or the average distance to the K-nearest neighbors.

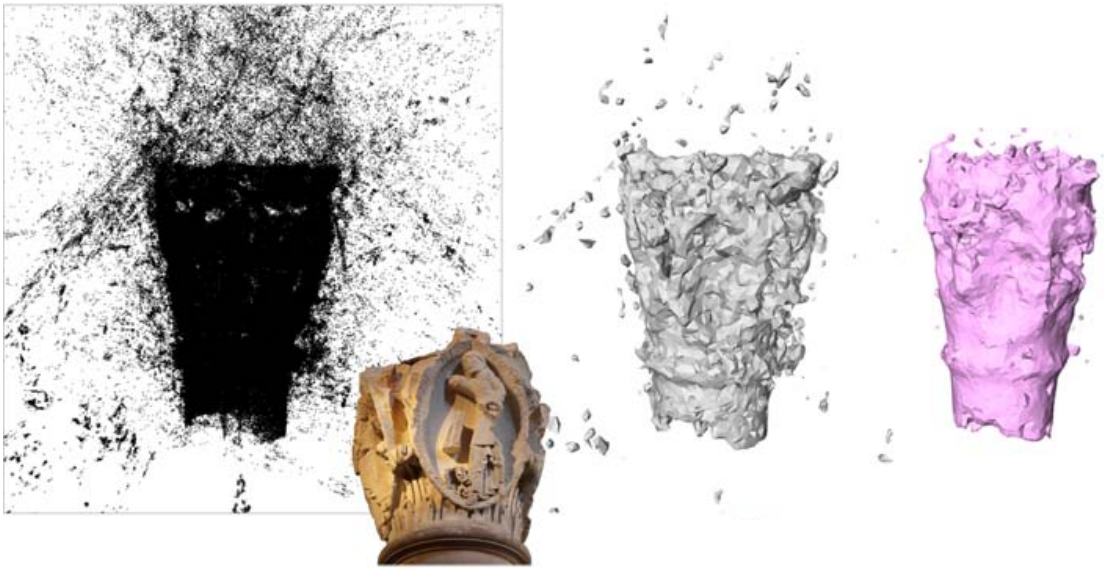
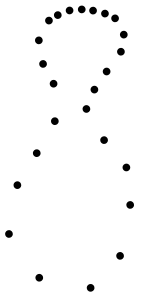


Figure 2.9: **"Signing the unsigned" algorithm [64].** Left: input point cloud and photo. Middle: estimation of the inferred surface. Right: reconstructed surface.

Removing outliers as a preprocessing step of reconstruction is another option but requires a trial-and-error process to adjust the parameters. Research has lately focused on specific distances both robust to outliers and suitable for reconstruction.

2.3.3 Non-Uniform Sampling Density



Isolated points may not be outliers but instead belong to a very low sampled surface: dealing with these points calls for different approaches.

Radial Basis Functions

Radial Basis Function (RBF) is one of the few reconstruction approaches with low dependence to the sampling density. A RBF is a radially symmetric function whose value depends on the distance from a given origin. A weighted sum of m RBF with centers c_i is used to define a function f at query point $\boldsymbol{q}$:

$$f(\boldsymbol{q}) = \sum_{i=1}^{m} w_i \theta(\|\boldsymbol{q} - \boldsymbol{c_i}\|).$$

Considering the input point cloud as discrete realizations of an implicit function, the reconstruction problem boils down to finding a set of primitive RBF and their associated weights. Carr et al. [21] consider the input as a set of RBF centers and search for an implicit function that is zero at the input points. The variables solved for are the weights of the RBF. To avoid the trivial solution of an implicit function that is zero everywhere, some points with non-zero values are added both inside and outside of the inferred shape, in accordance to to the oriented normals provided as input.

A formulation of the RBF system of equations is given:

$$\begin{pmatrix} A & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix},$$

where $A_{i,j} = \theta(\|\boldsymbol{p_i} - \boldsymbol{p_j}\|)$ and $P_{i,j}$ are low degree polynomials. Common RBF functions $A_{i,j}$ are biharmonic splines $\theta(r) = r$ (where r is the radius) or triharmonic functions $\theta(r) = r^3$.

Reconstruction from a point cloud with non-uniform sampling density is depicted by Figure 2.10.

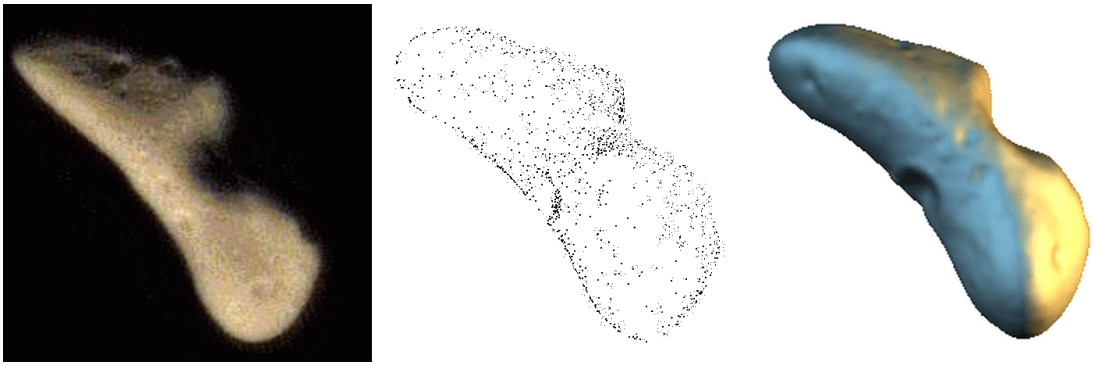


Figure 2.10: **Radial basis functions [21].** Left: photo of asteroid Eros. Middle: input point cloud with variable sampling density. Right: reconstruction using RBF.

One limitation of this method is its computational complexity: all the input points are taken into account when computing a single RBF value, resulting in a dense system of equations.

To overcome this limitation, variants have been proposed such as Compactly Supported RBFs [66], trading a loss of globality for an increased efficiency. However, the size of the supports of these CSRBF must be user-specified, which makes this approach ill suited to deal with widely variable sampling.

In order to remain robust to imperfect sampling while decreasing the overly high complexity, RBF can be handled through Multilevel Partition of Unity (MPU) [65]. A fitting function is associated to each RBF computed on a space subdivided by an octree structure. Separated contributions of RBFs with compact support are blended on local neighborhoods of the octree to compensate for the loss of globality.

Algebraic Point Set Surfaces

A variant of the Point Set Surfaces discussed in Section 2.3.1 has been contributed by Guennebaud and Gross [41]. It extends the PSS framework by replacing planar fitting by algebraic sphere fitting.

An algebraic sphere is defined as the 0-isosurface of the scalar field $s_{\mathbf{u}}(\mathbf{x}) = [1, \mathbf{x}^T, \mathbf{x}^T\mathbf{x}]\mathbf{u}$, where $\mathbf{u} \in \mathbb{R}^{d+2}$ denotes the vector of scalar coefficients describing the sphere. For $u_{d+1} \neq 0$, the center $\mathbf{c}$ and radius r of the sphere is given by:

$$\mathbf{c} = \frac{1}{2u_{d+2}}[u_0, ..., u_d]^T,$$

$$r = \sqrt{\mathbf{c}^T\mathbf{c} - u_0/u_{d+1}}.$$

The algebraic sphere fit at query point $\mathbf{q}$ minimizes the following energy:

$$E(\mathbf{q}) = \|\mathbf{W}^{\frac{1}{2}}(\mathbf{x})\mathbf{D}\mathbf{u}\|^2,$$

where:

- $\mathbf{W}$ is a $n \times n$ weight matrix;
- $\mathbf{D}$ is the $n \times d + 2$ matrix with its i^{th} line being $[1, \mathbf{p}_i^T, \mathbf{p}_i^T\mathbf{p}_i]$.
- $\mathbf{u}$ is constrained to avoid the trivial solution.

Note that this fitting step does not require any normal vector information, while another energy formulation deals with normal vectors ([41], section 4.3).

The Algebraic PSS is defined as the zero set of the implicit scalar field representing the algebraic distances between a query point $\mathbf{q}$ and its fitting sphere $\mathbf{u}(\mathbf{x})$.

Algebraic sphere fitting has proven extremely robust to low sampling rate compared to plane fitting which requires a large and consistent neighborhood. A comparison is provided by Figure 2.11.

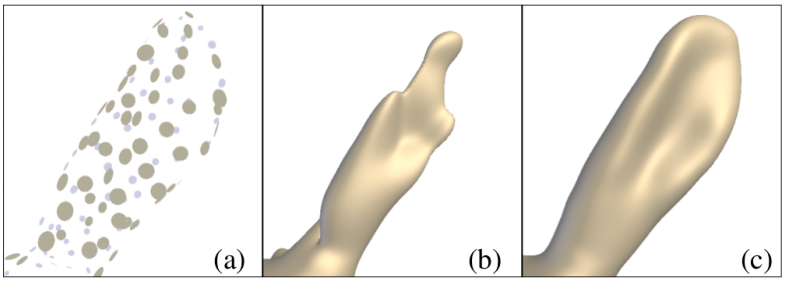


Figure 2.11: **Algebraic point set surfaces [41].** a. Undersampled input. b. Plane fitting PSS. c. Sphere fitting PSS.

GlobFit

The approaches relying upon primitives robust to variable sampling density.

As discussed in Section 2.1, these methods take advantage of strong assumptions made for the output. It may consist in a set of simple canonical primitives (planes, balls, cylinders, etc.), but can also be constrained to a collection of complex shapes (urban furnitures, mechanical pieces, etc.).

The method called GlobFit [57] is based on the RANSAC algorithm [74] for shape detection: planes, cylinders and balls are robustly detected inside the point cloud and their mutual relations are then estimated.

Each point $\boldsymbol{p_i}$ of the input point cloud $\mathcal{P}$ is equipped with a normal $\boldsymbol{n_i}$ and a confidence score w_i computed by local covariance analysis. The algorithm is initialized by partitioning the input into small subsets with associated primitives and some remaining isolated points.

A relationship graph is then constructed by estimating orthogonal or parallel relations: this is achieved by comparing the normals of pairs of primitives. Then, a maximal subset of relations is extracted, without any redundant constraint and any conflict.

In order to minimize the approximation error while conforming to the constraints extracted, a nonlinear optimization is performed over the parameters of the primitives. The

input points that are inliers of the current set of optimized primitives are identified and the algorithm is repeated for the remaining isolated points.

This algorithm has shown to be robust to noise, outliers and non-uniform sampling density (Figure 2.12). Robustness to variable sampling density mainly comes from the RANSAC algorithm: in this framework, the confidence of a primitive is evaluated by computing the approximation errors from the estimated inliers to the primitive, with no dependence to the local density. Primitives such as plane have infinite area, a measure of density of a plane is therefore meaningless.

Figure 2.12: **GlobFit algorithm [57].** Primitives are fit to a point cloud with non-uniform sampling density.

Overview

Because shapes cannot always be sampled with a constant regularity, the sampling density of acquired point clouds is rarely uniform. On some cases, this non-uniformity can become significant enough so that usual methods that assume regular sampling fail in producing a reliable reconstruction.

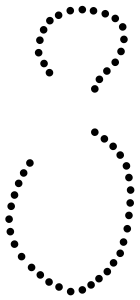
There are several ways to achieve robustness to non-uniform sampling density: primitive-based reconstruction methods, for example, use strong assumptions on the output shape so that the quality of a fit does not depend on the sampling density.

From a more computational point of view, their exists metrics that are able to variably weight the contribution of points according to their local density. In the RBF framework,

the sizes of the functions centered on the input points are unknown: they are computed to minimize a spline energy.

Finally, some methods like APSS derive from other usual reconstruction methods, and extend them to metrics that are more stable when density decreases significantly. This is very effective even in the case of very sparse sampling.

2.3.4 Missing Data



The ways of handling missing data strongly depend on the purpose of reconstruction: it can be considered as an extreme case of non-uniform sampling, where some parts of the scene have zero density. In this case, a surface may still be produced by interpolating from the non-empty areas of the scene while matching a regularization assumption such as smoothness, minimal area, etc.

Cone Carving

One relevant information to reconstruct a shape in the presence of missing data is visibility. Missing data is often produced by occlusions where an object hides a part of another one, and visibility vectors may help distinguish missing data from actually empty parts of the scene.

Shalom et al. [75] introduced the notion of "cone carving". Cones of visibility, estimated to not intersect the inferred surface, are computed from each point. In 2D, a visibility cone is computed as the largest section of the domain defined by two rays originated at input point $\boldsymbol{p_i}$ which does not contain any input point. In 3D, it is an empty generalized cone with apex at input point $\boldsymbol{p_i}$ and formed by all the rays oriented outward from the shape.

In 3D, a coarse approximation of the surfaces is needed to estimate the visibility cones. A silhouette is computed by connecting splats computed around the input points so that thin cones between neighbor sample points are discarded. The union of the cones is an estimation of the outside of the shape as depicted by Figure 2.13.

This method outperforms for example in the case of a surface parallel to the line of sight of the scanner and thus sparsely sampled. Most variational methods would create a bump near the boundary of missing data (minimizing, for instance, the global curvature).

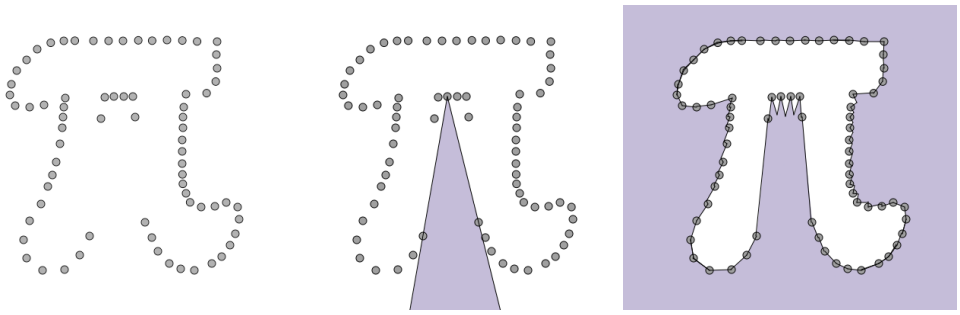


Figure 2.13: **Cone carving algorithm [75].** Left: input with missing data. Middle: one visibility cone depicted. Right: union of visibility cones.

Cone carving instead infers the correct plane by using the visibility of the points on the boundary of missing data.

For shape reconstruction purposes, a union of cones may be combined with the RBF approximation: it strongly constraints the inside and outside of the inferred shape. The main weakness of this method is that it requires a splatting step: a local neighborhood must be selected, which requires a trial-and-error process.

Visibility-Aware Surface Evolution

To achieve robustness to missing data, Tagliasacchi et al. [79] seek for a trade-off between visibility and some smoothness parameters. The "VASE" approach (*Volume-Aware Surface Evolution*) initializes a 2-manifold surface S as a scaled bounding box of the input point cloud. The surface then evolves according to the following energy:

$$E = \omega_1 E_{fit} + \omega_2 E_{smooth} + \omega_3 E_{vol},$$

where:

- E_{fit} measures data fidelity while respecting the visibility vectors $\boldsymbol{v_i}$ at points $\boldsymbol{p_i}$;
- E_{smooth} favors surface smoothness;
- E_{vol} favors volumetric smoothness;
- $\omega_{1,2,3}$ are relative weights for each energy.

When defining the evolving implicit surface S_ϕ^t with ϕ a signed distance field and t the iteration count, the energy is reformulated through level-sets:

$$\check{\phi} + \omega_1 F_{fit} + \omega_2 F_{smooth} + \omega_3 F_{vol}.$$

The energy terms become forces and are defined as follows:

$$F_{fit} = \delta(\phi) \sum_{i=1}^{n} \varpi(v_i^{\parallel}) v_i^{\parallel} f(v_i^{\perp}); \quad F_{smooth} = -\Delta_S \kappa; \quad F_{vol} = -\Delta_M R;$$

where:

- $\delta(\phi)$ denotes the Dirac function.
- f denotes a Gaussian function with kernel width proportional to sampling density.
- $v_i^{\parallel}$ and $v_i^{\perp}$ are the parallel and normal components of $\boldsymbol{v_i}$ with respect to the normal vector of S at this location.
- ϖ is a function controlling the depth of the region behind the sample where the visibility vector impacts the surface.
- Δ is the standard Laplacian operator.
- κ is the mean curvature of the surface.
- M is a structure representing the surface (*Medial Axis Transform* [76]).
- R is a radius function defined on M, providing a volumetric representation of the surface: variations of R along M indicates variations in the local volume of the shape.

The rationale behind this trade-off between 3 forces is that each sample point is visible from the scanner and that the volume of a shape varies smoothly. Figure 2.14 shows a sum-up of this algorithm.

This method provides a smoother control of the visibility weight through the use of the scalars $\omega_{1,2,3}$. It can also be noted that the information about the volume that is surrounded by the inferred surface is a noticeable help for reconstruction.

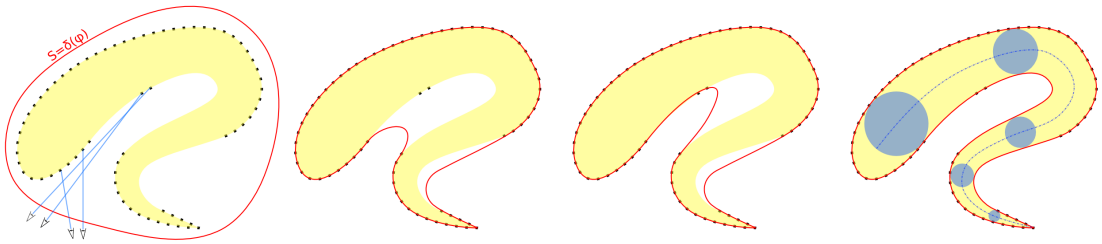


Figure 2.14: **VASE algorithm [79].** From left to right: initialization; reconstruction using only smoothness and point constraints; impact of adding visibility direction constraint; VASE reconstruction.

Medial Kernels

The main challenge posed by missing data is the estimation of the *shape* of a missing part of the scene.

Berger et Silva [12] estimate it by using medial balls. A medial ball is a ball which is equidistant and tangential to at least two surface points, maximally empty and inside of the shape. They aim at computing maximal connected components from a set of estimated medial balls.

Given two points $\boldsymbol{p_i}$ and $\boldsymbol{p_j}$ in the input $\mathcal{P}$ with associated normals $\boldsymbol{n_i}$ and $\boldsymbol{n_j}$, a candidate ball is generated, equidistant to $\boldsymbol{p_i}$ and $\boldsymbol{p_j}$ and whose normals at the points are similar to $\boldsymbol{n_i}$ and $\boldsymbol{n_j}$.

The center $\boldsymbol{c_{i,j}}$ of the ball lies on the bisecting plane of the two points. For the normals to coincide with $\boldsymbol{n_i}$ and $\boldsymbol{n_j}$, lines formed by them with the points are intersected with the bisecting planes at points $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$. To avoid arbitrary large balls that can be produced around sharp features, the center $\boldsymbol{c_{i,j}}$ is chosen, between $\boldsymbol{x_i}$, $\boldsymbol{x_j}$ and their midpoint, to be the one producing the minimal radius $r_{i,j}$.

If either intersection is along the positive direction of their normals, the ball is lying in the exterior of the shape and is discarded: such idea leads to major improvement when dealing with missing data close to thin holes and nearby surface sheets, as shown on Figure 2.15.

To compute the deviation of the candidate ball from a medial ball, two measures are defined:

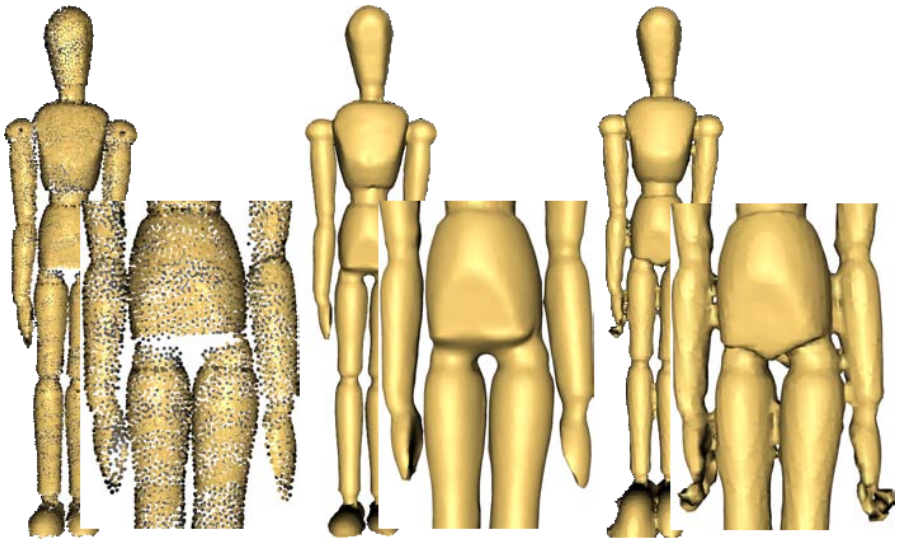


Figure 2.15: **Medial kernels [12].** Left: input point cloud. Middle: medial kernels reconstruction. Right: RBF reconstruction [66].

$$\gamma(\boldsymbol{p_i}, \boldsymbol{p_j}) = \sum_{p \in \mathcal{P}} \mu(\boldsymbol{c_{i,j}}, r_{i,j}, \boldsymbol{p}),$$

$$\tau(\boldsymbol{p_i}, \boldsymbol{p_j}) = |\boldsymbol{n_i} - \boldsymbol{s_i}| + |\boldsymbol{n_j} - \boldsymbol{s_j}|,$$

where $\boldsymbol{s_i}$ and $\boldsymbol{s_j}$ denote the normals of the candidate ball at points $\boldsymbol{p_i}$ and $\boldsymbol{p_j}$ and μ is a function measuring how close a point $\boldsymbol{p}$ lies from the center of the ball $\boldsymbol{c_{i,j}}$. Here, γ measures the emptiness of the ball whereas τ measures how tangential it is.

From these two measures, a similarity measure is defined:

$$\phi(\boldsymbol{p_i}, \boldsymbol{p_j}) = \exp\left(-\left(\frac{\gamma(\boldsymbol{p_i}, \boldsymbol{p_j})}{\sigma_e}\right)^2 - \left(\frac{\tau(\boldsymbol{p_i}, \boldsymbol{p_j})}{\sigma_t}\right)^2\right),$$

where σ_e and σ_t define bandwidths for the emptiness and tangential measures.

This measure is computed for each pair of points to fill a similarity matrix $M : M_{i,j} = \phi(\boldsymbol{p_i}, \boldsymbol{p_j})$. Each cell encodes the likelihood of two points to belong to the same medial ball. Most coefficients being close to zero, the matrix is quite sparse.

Performing spectral analysis of this matrix allows for volume-aware segmentation and reconstruction by parts.

Overview

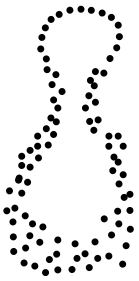
Missing data is an enduring problem in shape reconstruction. Reconstruction is ill-posed and multi-faceted: should we only reconstruct the scanned part of a shape or should we compensate for the lost information? To what extend? Based on which assumptions?

Primitive-based method usually handle it the same way they handle variable sampling, that is to say by only measuring data fidelity to a predefined shape. In general, variational methods handle it via their smooth closed prior. As the output is by definition a closed shape, a hole in the data is filled to close the reconstructed shape. Albeit robust, this approach is naive as the mathematical assumption used to fill the holes (minimal surface or curvature, for instance) does not necessarily match the ground truth. To provide better guidelines to fit reality, some techniques were devised, taking for example visibility into account. This prevents the shape from crossing parts of the scene that are known to be empty because of the relative positions of the acquisition device and of the input points.

Visibility provides crucial knowledge about what belongs to the outside of the shape. Other approaches such as medial kernels [12] increase knowledge about the inside by the means of statistics, computing the relative probabilities that points belong to the same local part of the shape.

This recent class of methods are a significant improvement in robustness to missing data, but can only handle point clouds with oriented normal vectors, which is a strong requirement for the input.

2.3.5 Variable Noise



Handling variable noise is substantially different from handling a constant level of noise, because it cannot be done through a global analysis of the scale. The automatic handling of noise is commonly achieved through bandwidth selection, which means adapting a scale parameter to the level of noise.

Bandwidth Selection for MLS

Wang et al. [82] extend the use of MLS by computing optimal bandwidths.

As detailed in Section 2.3.1, the MLS framework requires a local reference domain usually defined as a plane:

$$H = \arg\min \sum_{i=1}^{n} (\langle \boldsymbol{n}, \boldsymbol{p_i} \rangle - D)^2 \theta(\|\boldsymbol{p_i} - \boldsymbol{q_H}\|).$$

The weight function θ is the scale of the MLS, referred to as *bandwidth*.

When H is found, a polynomial approximation of the surface is sought after:

$$g = \arg\min \sum_{i=1}^{n} (g(x_i, y_i) - f_i)^2 \theta(\|\boldsymbol{p_i} - \boldsymbol{q}\|),$$

where (x_i, y_i) denote the coordinates of the projection of $\boldsymbol{p_i}$ in the local coordinate system of H, and f_i is the distance of $\boldsymbol{p_i}$ from H.

A commonly used scale function is the Gaussian, possibly truncated $\theta(r) = e^{-\frac{r^2}{h^2}} \varphi(r)$, where φ is a windowing function used to increase computational efficiency.

The bandwidth is set by the parameter h. A kernel regression is performed to automatically select it. The optimal bandwidth is defined as the one minimizing the mean squared error between a point and the projection operator. This leads to the following expression:

$$h_{opt} = C \left(\frac{v}{n\rho(\boldsymbol{q})(f''(\boldsymbol{q}))^2} \right)^{1/5},$$

where:

- C is a constant dependent on the kernel.
- v is the variance of the noise.
- $\rho(\boldsymbol{q})$ is the local density of points at query point $\boldsymbol{q}$.
- $f(\boldsymbol{q})$ is the projection operator applied at query point $\boldsymbol{q}$.

The optimal scale is selected as the one minimizing the error of a kernel regression problem, allowing for reconstruction of both noise-free and noisy point clouds without any user-defined setting (Figure 2.16).

Note that even though automatic, this scale selection remains global hence non-adaptive.

Statistical Estimator

Inferring the scale of a point cloud is commonly handled through minimizing an approximation error.

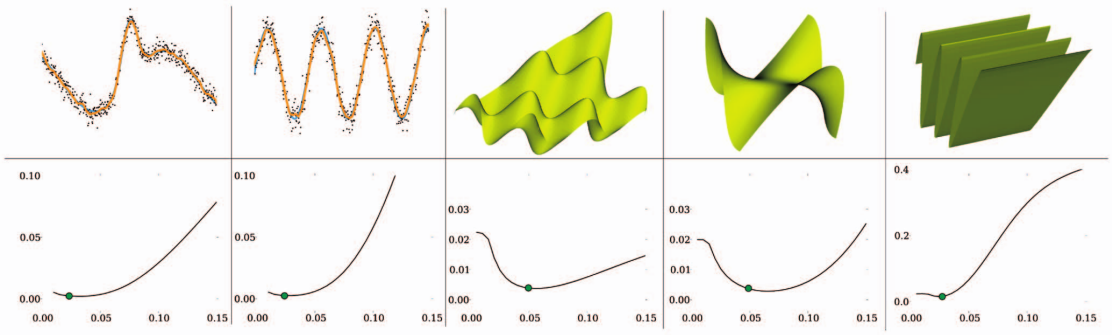


Figure 2.16: **Optimal bandwidth selection [82].** A kernel regression is performed on each point cloud. The associated functional (and minimum chosen) is given under each example.

Unnikrishnan et al. [80] investigate the variation of a statistical estimator where an input point is considered as a noisy observation of a real point lying on the inferred shape. A semi-parametric model is used: the shape is smooth with parameters κ (curvature) and τ (torsion), both close to constant on a small neighborhood. The noise is assumed Gaussian with a variance of σ^2.

The estimator for sample covariance matrix can be expressed as the sum of the matrix of its expected value and a matrix of random variables $\hat{M}_n = \bar{M} + Q$, where Q denotes a symmetric perturbation matrix. The impacts of varying Q over principal eigenvector of $\hat{M}_n$ are computed and analyzed. The support radius size is adjusted to estimate the most reliable model parameters.

As depicted by Figure 2.17, the optimal scale can for example be considered as the one minimizing the error on the estimated tangent plane of a local subset of the input point cloud.

The statistical analysis of variable noise models provides satisfying results but requires strong assumptions on the input (noise model, smoothness).

Growing Least Squares

Mellado et al. [61] tackled the problem of variable noise by defining an *adaptive bandwidth*. A random point $\boldsymbol{q}$ at any scale s is characterized by a low-degree algebraic surface that best approximates its neighborhood $Q_s = \{\boldsymbol{p_i}; \|\boldsymbol{p_i} - \boldsymbol{q}\| \leq s\}$. Each input point $\boldsymbol{p_i}$ is equipped with a normal vector $\boldsymbol{n_i}$.

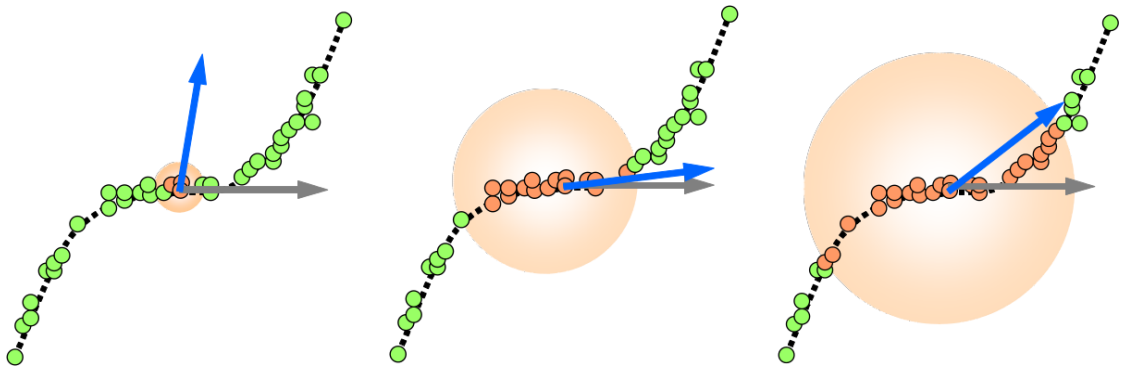


Figure 2.17: **Statistical estimator [80].** Left: scale is too small, the plane is randomly tangent to the noise. Middle: correct scale, the plane is close to the real tangent plane. Right: scale is too large, the curvature enters into play and deviates the plane.

The Algebraic PSS framework is used (2.3.3): algebraic spheres are fitted as 0-isosurfaces of the scalar field $s_{\mathbf{u}}(\mathbf{x}) = [1, \mathbf{x}^T, \mathbf{x}^T\mathbf{x}]\mathbf{u}$ with $\mathbf{u} = [u_c\, \mathbf{u_l}\, u_q]^T$ a vector of parameters.

$\mathbf{u}_l$ and u_q are computed by minimizing $\sum_i w_i(s)\|\Delta s_{\mathbf{u}}(\boldsymbol{p_i}) - \boldsymbol{n_i}\|^2$. Then, the constant coefficient u_c is obtained by minimizing $\sum_i w_i(s)\|s_{\mathbf{u}}(\boldsymbol{p_i})\|^2$.

The weights w_i are scale-dependent:

$$w_i(s) = \left(\frac{\|\boldsymbol{p_i} - \boldsymbol{q}\|^2}{s^2} - 1\right)^2.$$

A geometric scale-invariant descriptor $D = [\tau, \eta, \kappa] =$ is defined for a query point $\boldsymbol{q}$ and a scale s:

$$\tau = s_{\hat{\mathbf{u}}}(\boldsymbol{q})\,; \quad \eta = \frac{\nabla s_{\hat{\mathbf{u}}}(\boldsymbol{q})}{\|\nabla s_{\hat{\mathbf{u}}}(\boldsymbol{q})\|}\,; \quad \kappa = 2\hat{u}_q.$$

This three measures represent respectively the algebraic offset distance, the unit normal of scalar field and the signed curvature of the hyper-sphere. The main rationale behind the automatic bandwidth selection is to analyze the variation of the geometric parameter:

$$\nu(\boldsymbol{q}, s) = \left(\frac{\mathrm{d}\tau}{\mathrm{d}s}\right)^2 + \left(s\frac{\mathrm{d}\eta}{\mathrm{d}s}\right)^2 + \left(s^2\frac{\mathrm{d}\kappa}{\mathrm{d}s}\right)^2.$$

At query point $\boldsymbol{q}$, the scale s is decreased until this variation $\nu(\boldsymbol{q}, s)$ exceeds a user-defined threshold (Figure 2.18). This pioneered the problem of multiple levels of noise on

a single scene, by defining a scale as a function of the position $\boldsymbol{q}$ in space.

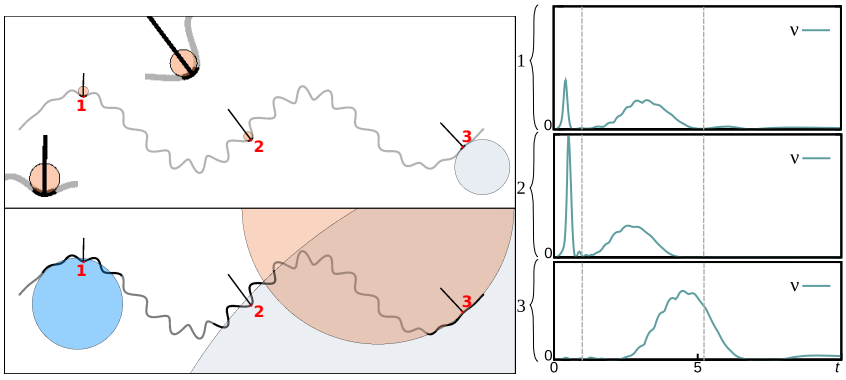


Figure 2.18: **Adaptive bandwidth selection.** Multiple scales can be selected by the analysis of ν, a geometric variation function.

Overview

Robustness to variable noise is still an open problem and a recent topic of interest in shape reconstruction from measurement data.

Automatically selecting the scale is a significant step towards these goal. Adapting automatically to noise to is commonly achieved through varying the scale and analyzing its impact over one or several measurements: for example, the quality of the fit of a tangent plane is more likely to be high if the scale does not exceed the local planarity of a point cloud.

Such notion of a scale that locally adapts to the properties of input subsets is ill-posed: the analysis of the scale is performed on a local subset whose size depend on the local scale.

the scale must be put to the test on a local subset, but the size of this local subset depends on the chosen scale.

The rationale of the adaptive bandwidth is to lower the scale away from the global one until the subset geometric properties lose stability.

2.4 Conclusion

The state of the art in smooth closed shape reconstruction is as wide and diverse as the input point clouds and requirements for the output shapes. Methods designed to handle perfect data sets often provide proven mathematical guarantees. Approaches seeking robustness to defects, on the other hand, often approximate the point cloud under specific requirements.

Noise robustness has been tackled from a variety of viewpoints but has become, in the last few years, more focused on automatic scale selection, motivated by the need to get rid of the user-defined parameters which often require a trial-and-error process. Robustness to variable levels of noise remains an open problem and the selection of adaptive scales a recent topic of interest.

Outliers and non-uniform sampling density are different but not unrelated defects. Isolated points are a product of both of these two defects, and it seems difficult to automatically distinguish them. While a series of approaches initially tackled this problem through explicit formulations like outlier filtering, the recent trend is to use metrics or operators that are inherently robust to these defects.

Missing data can be seen as a specific case of variable sampling. Many methods handle it implicitly by "filling the holes" as they only produce closed shapes. Robustness to missing data has become a topic of interest as it touches the even more ill-posed problem of data completion. Estimating the shape of a missing part of a scene calls for specific ways of considering the input point cloud, from a visibility point of view for example.

3

Problem Statement

We consider an input point cloud with no additional measurement: neither normal vector, nor scale or visibility information. It might be ridden with the following defects: noise (possibly with variable magnitude), outliers and missing data.

Points in undersampled regions being not distinguishable from outliers, our method is not robust to variable density.

In this part, we focus on a specific subset of possible output shapes.

Smooth

The output shape does not contain any sharp feature and is smooth everywhere. A scanned shape with sharp features is handled with this method, at the cost of a loss of sharpness of its features (rounded corners, for example).

Closed

The output shape is defined as the interface between an inside domain and an outside domain. It surrounds a 2D domain, in the case of curve reconstruction in 2D, or a volume, in the case of surface reconstruction in 3D.

This requirement also applies to the input to some extend. A point cloud sampling an opened shape may be reconstructed as a closed shape if the inferred shape contains

a strong concavity, that is to say if an inside can still be clearly distinguished from an outside. Likewise, a closed shape scanned with missing data is also handled.

Figure 3.1 shows plausible reconstruction from a variety of input point clouds.

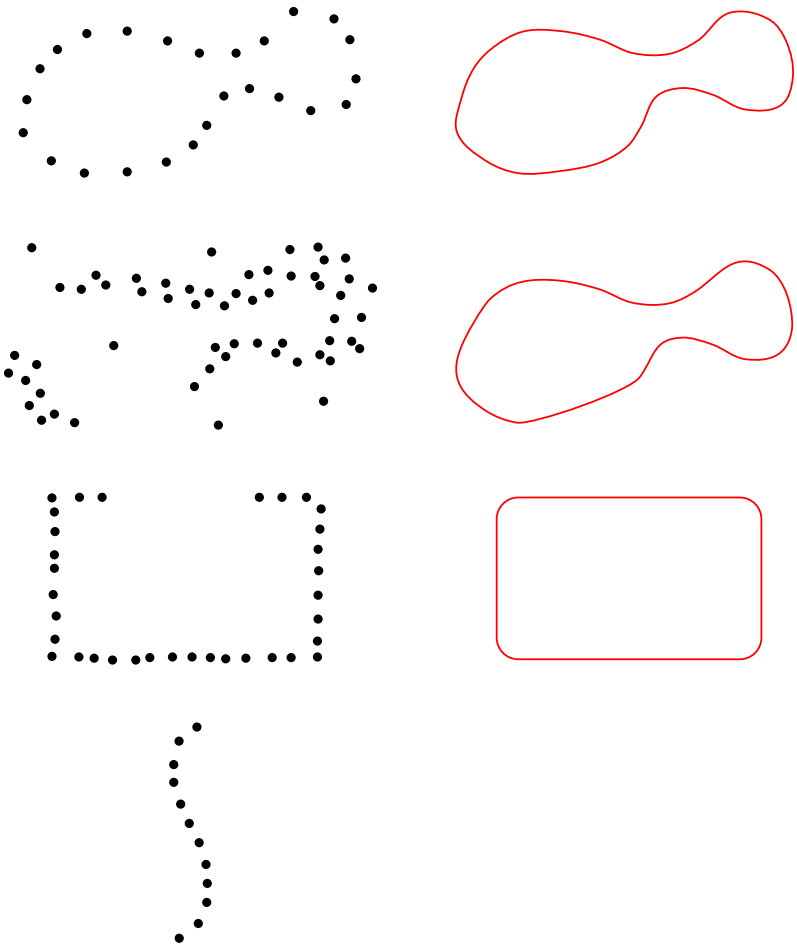


Figure 3.1: **Plausible reconstructions.** Left column: input point cloud. Right column: plausible smooth closed reconstruction. The last point cloud does not sample the border of a domain and cannot be reconstructed.

4

Background

4.1 A priori Knowledge

4.1.1 Dimension

Shape reconstruction from raw point clouds is an ill-posed problem. First, the dimension of input an output differ. An infinity of smooth curves of surfaces could "fit" the input point cloud. As no additional measurements are given, we need to regularize the problem by relying upon one or several assumptions about the inferred shape.

We denotes by s the global scale of the scene, used to match a subset of the input subset of specific size with a part of the output. This scale is correlated to the level of noise of the scene. It is usually defined either by setting the cardinality or the volume of a neighborhood of a subset:

- **A number of points K:** from a query point $\boldsymbol{q}$, the K nearest neighbors are considered as witnesses of the inferred shape close to $\boldsymbol{q}$.
- **A radius σ:** from a query point $\boldsymbol{q}$, the points lying inside a ball $B(\boldsymbol{q}, \sigma)$ of radius σ are considered witnesses of the inferred shape close to this point.

In our context, the scale refers to the number of neighbor points. Our *a priori* knowledge about dimension is that a subset of size K of the input point cloud should have an apparent dimension of a curve in 2D or a surface in 3D, depending on the inferred object.

4.1.2 Geometry

To make the problem better-posed, we assume that the inferred shape can be represented as an isolevel of an implicit function f defined all over the domain. Adding such implicit formulation, the problem is subdivided in two steps:

1. Compute an implicit function from a set of points.
2. Extract a curve in 2D or a surface in 3D from an isovalue of this function.

The second step, going from a function f to a reconstruction is commonly handled by extracting an isolevel of the function. More specifically, the output shape is defined by $f^{-1}(c)$, i.e., the locus where the function evaluates to a constant c. This constant is commonly chosen be chosen to be 0 for a signed implicit function or the mean value of f at the input points. We choose to use the median value that the function f takes on the input points for outlier robustness.

The first step, computing the implicit function is our core problem. As we saw in Section 2.3.1, implicit functions are widely used for reconstruction: the Poisson reconstruction [48] approach solves for an indicator function whose gradient best fits, in the least square sense, the normal vectors provided as attributes to the input points. We only require raw point clouds and wish to offer robustness to noise and outliers. We describe next a robust distance function that matches these requirements.

4.2 Optimal Transportation

4.2.1 Formulation

The transportation theory tackles the problems of optimal transportation and allocation of resources. The transportation problem can be intuitively described as follows. Considering

a pile of sand on a domain μ and a hole of the same volume in a domain ν, what is the most effective way to move the sand so as to fill the hole?

If we assume the sand is moved one infinitesimal unit of volume at a time, we can define a point-to-point mapping referred to as the *transport plan*. The effectiveness of a transport plan is measured as the sum of the distances for each grain of sand.

This formulation is referred to as Monge's variational formulation. It was extended by Kantorovich to transport plans between probability measures. Considering two probability measures μ and ν, the transport plan is a probability measure π on $X \times Y$ whose marginals are μ and X and ν on Y. For probabilities with finite support, $\pi(x, y)$ corresponds to the amount of mass transferred from x to y.

This problem strongly depends on the definition of "effectiveness", that is to say the *transport cost*. A standard cost function is the q-Wasserstein metric w_q defined as:

$$w_q(\mu, \nu) = \left(\inf_{\pi} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|x - y\|^q d\pi(x, y) \right)^{\frac{1}{q}}.$$

In the context of the sand transportation problem, this is the cost of moving the sand to the hole with the L_q distance. In our context, optimal transport formulation has two main advantages:

- It is applicable to point clouds, by considering a point as a Dirac mass.
- A distance function can be defined everywhere: this is a first step towards an implicit function.

4.2.2 Robust Distance Function

Definition

Chazal et al. [23] introduced the notion of a robust distance function to a measure.

Consider a measure μ with finite second moment, i.e., with finite support on its domain of definition. The distance function from a query point x to μ with parameter m_0 is defined as the transport cost in the 2-Wasserstein sense:

$$d^2_{\mu,m_0} : \mathbb{R}^d \to \mathbb{R}, \ \boldsymbol{x} \mapsto \frac{1}{m_0} \int_0^{m_0} \delta_{\mu,m}(\boldsymbol{x})^2 \mathrm{d}m.$$

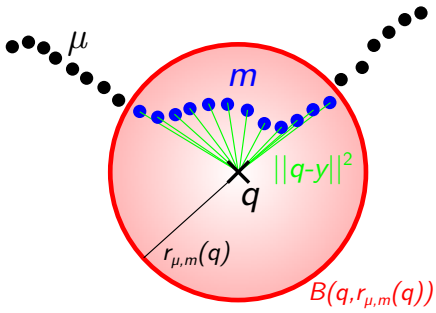


Figure 4.1: **Robust Distance Function from a query point q to a measure μ.**

Taking for $\delta_{\mu,m}(\boldsymbol{x})$ the Euclidean distance from $\boldsymbol{x}$ to its nearest point in μ, the distance evaluates to:

$$d^2_{\mu,m_0} : \mathbb{R}^d \to \mathbb{R},\ \boldsymbol{x} \mapsto \frac{1}{m_0} \int_{B(\boldsymbol{x}, r_{\mu,m_0}(\boldsymbol{x}))} \|\boldsymbol{x} - \boldsymbol{y}\|^2 \mathrm{d}\mu(\boldsymbol{y}),$$

where $r_{\mu,m_0}(\boldsymbol{x})$ is the minimal radius r such that the ball centered at $\boldsymbol{x}$ with radius r encloses a mass of at least m_0. This is illustrated in Figure 4.1.

A point cloud may be seen as a distribution of Dirac masses. The robust distance function then simplifies to:

$$d^2_{\mu,K/n} : \mathbb{R}^d \to \mathbb{R},\ \boldsymbol{x} \mapsto \frac{n}{K} \sum_{i=0}^{K-1} \|\boldsymbol{x} - \boldsymbol{x_i}\|^2,$$

where K is the number of nearest neighbors of $\boldsymbol{x}$ such that $m_0 = K/n$ and $\boldsymbol{x_i}$ is the i^{th} nearest neighbor of $\boldsymbol{x}$.

Properties

- **Regularity:** a function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be $L - concave$ if the function $\boldsymbol{x} \mapsto L\|\boldsymbol{x}\|^2 - f(\boldsymbol{x})$ is convex.

  The square of the distance function is 1-semiconcave. It is differentiable almost everywhere in $\mathbb{R}^d$ and its gradient is:

$$\Delta_{\boldsymbol{x}} d^2_{\mu,m_0} = \frac{2}{m_0} \int_{h \in \mathbb{R}^d} [\boldsymbol{x} - \boldsymbol{h}] \mathrm{d}\mu_{\boldsymbol{x},m_0}(\boldsymbol{h}).$$

- **Stability:** the map $\mu \to d_{\mu,m_0}$ is $m_0^{-1/2}$-Lipschitz. This means that for two probability measures μ and ν on $\mathbb{R}^d$ and for $m_0 > 0$:

$$\|d_{\mu,m_0} - d_{\nu,m_0}\| \leq \frac{1}{\sqrt{m_0}} W_2(\mu, \nu).$$

- **Convergence:** let μ be a probability measure with compact support S and $m_0 > 0$. The distance function d_{μ,m_0} converges uniformly to the distance to S as m_0 goes to 0.

Reconstruction

From the three properties listed above, it has been shown that the sublevel sets of the distance function of a point cloud ridden with noise and outliers converges uniformly to its support ([23], Section IV.1). We detail next a shape reconstruction algorithm which takes advantage of this property.

4.2.3 Shape Reconstruction

Mullen et al. [64] proposed a shape reconstruction algorithm based on the robust distance function:

$$d_K : \mathbb{R}^d \to \mathbb{R},\ \boldsymbol{x} \mapsto \sqrt{\frac{1}{K} \sum_{\boldsymbol{p} \in N_K(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{p}\|^2},$$

where $N_K(x)$ is the set of K nearest neighbors to $\boldsymbol{x}$.

The algorithm consists in signing the robust distance function in order to define an implicit signed distance function whose 0-isovalue is a reconstruction of the inferred shape. Note that signing the robust distance function offers robustness to missing data by estimating the inside from the outside of the inferred shape. We summarize below the main steps of this approach.

Scale Estimation

To compute the distance function, the user specifies a number of nearest neighbors K. As the sublevel sets of the distance function are known to provide a good approximation

of the shape, a band defined by the domain where $d_K(\boldsymbol{x}) < \varepsilon$ is assumed to contain the inferred shape.

To estimate an adapted isovalue, an empirical function of ε is defined to analyze both the topology of the ε-band and the density of input points inside of this band:

$$M(\varepsilon) = \frac{C(\varepsilon) + H(\varepsilon) + G(\varepsilon)}{D(\varepsilon)},$$

where C, H and G denote the number of components, cavities and tunnels in the ε-band, and D is the density of input points in the ε-band.

The value for ε is empirically chosen as the first local minima reached by $M(\varepsilon$ after the first peak in the curve. The first peak reflects the high complexity of the ε-band topology when ε is too small. This value of ε allows to capture the correct topology while keeping the band thin and dense with points, as shown in Figure 4.2.

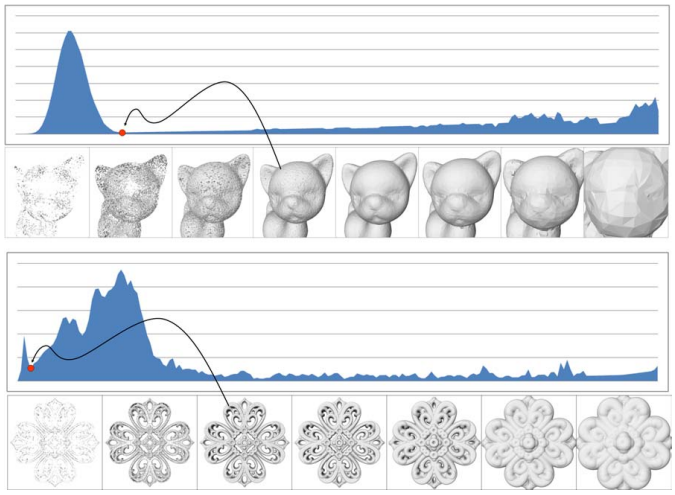


Figure 4.2: **Automatic ε-band width selection.** $M(\varepsilon)$ is plotted with corresponding bands shown underneath. The first local minimum after the first peak is the chosen ε value.

Function Computation

The robust distance function is computed on an octree covering the domain.

For quick nearest neighbors query, a k-d tree of the input point cloud is computed. The complexity of a query of K nearest neighbors in a point cloud of size n is $O(K \log(n))$.

Sign Guess

A parity test is perform in order to estimate whether a point lies inside or outside the shape. First, a ray is shot from the query point. Then, the unsigned distance function is computed along this ray. From this, we compute the parity of intersection between the ray and the ε-band. As the band is assumed surround the inferred shape, this provides a good estimate of the number of times the ray crosses the inferred shape. The point is estimated to be inside if the number of band-crossings is odd, outside otherwise.

Nevertheless, counting the number of band-crossing of a ray can lead to wrong estimation of the sign:

- **Missing data:** if the ray crosses a part of the shape missing from the input data, then the parity test is wrong;
- **Tangential ray:** as the ε-band has a non-zero width, a tangential ray might cross it but remain on the same side of the inferred shape. In this case, the parity test also fails (see Figure 4.3).

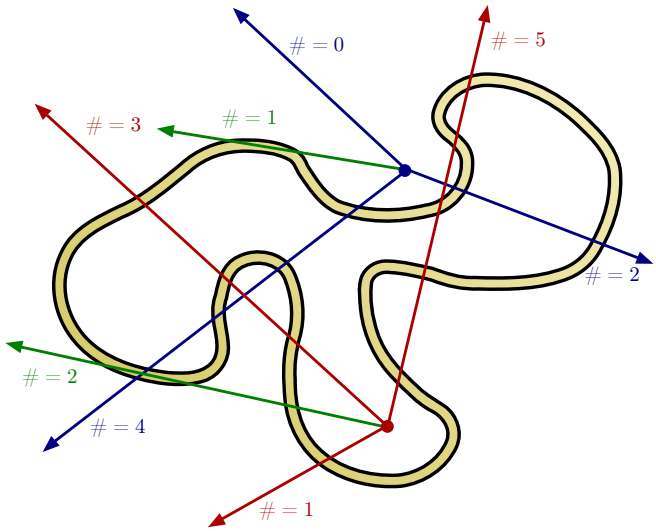


Figure 4.3: **Ray shooting.** Rays with correct parity detection are in **blue** (even) and **red** (odd). Tangential rays giving a wrong parity estimation are in **green**.

A stochastic approach is used to achieve both robustness to missing data and tangential

rays. Several different rays are shot from a single query point. This provides a sign guess with a certain confidence, from 1 if all parity tests agree to 0 if 50% of the rays are given an odd parity and 50% an even one. The sign is estimated at every point of the octree. A Laplacian operator is then applied to compute a smooth signed function, using both the robust distance function and the sign guess.

This method provides a fair amount of robustness for noise, outliers and to some extend missing data. One of its advantages is that it only requires raw point clouds, with neither normal vectors or visibility lines.

Several limitations remain however. First, the scale K is not automatically selected and thus must be user-defined. It is also global, hence non-adaptive. In addition, the sign is guessed through ray-shooting which becomes unreliable as the complexity of the scene increases and the number of potentially tangential rays grows.

5 Contribution

We present a novel reconstruction algorithm from defect-laden point clouds with variable noise and outliers. It comprises three main steps. First, we define and compute a noise-adaptive distance function to the inferred shape, which relies on a dimension assumption. We then estimate the sign and confidence of the function at a set of seed points. Finally, we compute a signed implicit function based on the noise-adaptive distance function and on the most confident seed points computed in previous steps.

5.1 Noise-Adaptive Distance Function

We propose a novel noise-adaptive distance function. For the sake of completeness, we go back to the continuous formulation of the distance function, using a mass m as a scale factor and a mass distribution μ to represent the input point cloud.

The noise-adaptive distance function is defined as follows:

$$\delta_{\mu,\alpha} = \inf_{m_0>0} \frac{d_{\mu,m_0}}{m_0^\alpha},$$

where α denotes a parameter that only depends on the dimensions of the space and of the inferred shape. The following subsection explains how this function is built.

5.1.1 Non-Adaptive Function

We first analyze the behavior and properties of the non-adaptive distance function in some ideal cases. Specifically, we analyze the effects of a change of scale m_0 for a fixed query point.

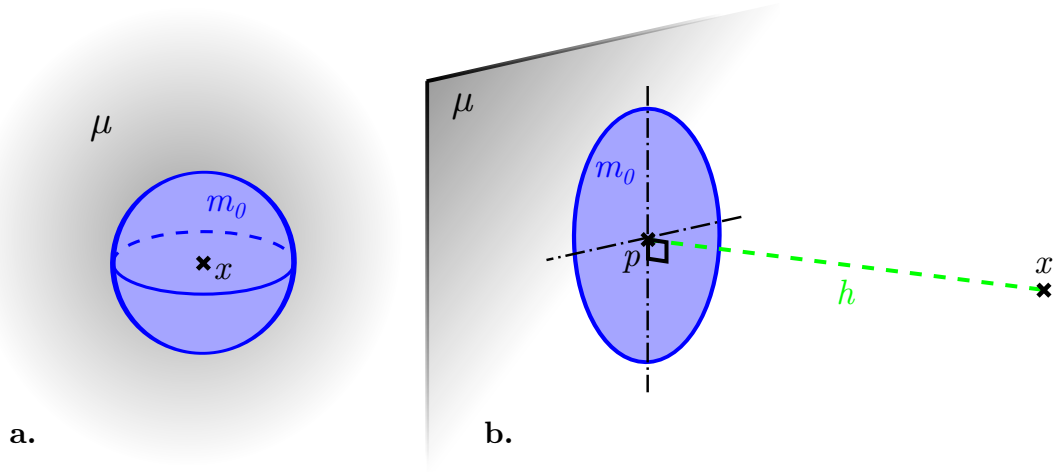


Figure 5.1: **Ideal cases for computing robust distance function. a.** ambient noise in 3D, the distribution μ is uniform on space and the mass m_0 is contained in a sphere. **b.** uniform measure μ on a 2-submanifold in 3D, the mass m_0 is contained in a disk on the supporting plane of μ.

Ambient Noise

Consider the case of an ambient noise, i.e., a uniform measure μ on a d-dimensional space, as shown on Figure 5.1-a. Remind that for any point $\boldsymbol{x} \in \mathbb{R}^d$, the robust distance function define as:

$$d^2_{\mu,m_0}(\boldsymbol{x}) = \frac{1}{m_0} \int_{B(\boldsymbol{x}, r_{\mu,m_0}(\boldsymbol{x}))} \|\boldsymbol{x} - \boldsymbol{y}\|^2 \mathrm{d}\mu(\boldsymbol{y}).$$

In this case, this value is constant on the whole space and does not depend on $\boldsymbol{x}$. The function is an integral on a ball of constant radius. To simplify notations, we replace $r_{\mu,m_0}(\boldsymbol{x})$ by R. The distance function simplifies to:

$$d^2_{\mu,m_0} = \frac{1}{m_0}\int_0^R r^2 \mathrm{d}\mu(r).$$

The volume of a d-ball of radius R is $C_d R^d$ with C_d a constant equal to $\frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}$. We call ρ_d the d-density of μ with $\rho_d = \frac{m_0}{CR^d}$, and apply this change of variable:

$$\mathrm{d}\mu(r) = \rho_d d C_d r^{d-1}\mathrm{d}r.$$

The expression of the robust distance function becomes:

$$d^2_{\mu,m_0} = \frac{\rho_d d C_d}{m_0}\int_0^R r^{d+1}\mathrm{d}(r) = \frac{d}{d+2}\frac{\rho_d C_d}{m_0}R^{d+2}.$$

By definition of ρ_d, we have $R = \left(\frac{m_0}{\rho_d C_d}\right)^{1/d}$. Thus:

$$d^2_{\mu,m_0} = \frac{d}{d+2}\frac{\rho_d C_d}{m_0}\left(\frac{m_0}{\rho_d C_d}\right)^{1+(2/d)} = \frac{d}{(d+2)(\rho_d C_d)^{2/d}}m_0^{2/d} = c_1 m_0^{2/d}.$$

This means that for a fixed query point, d_{μ,m_0} is varies proportionally to $m_0^{1/d}$. For a parameter $\alpha > 1/d$, $\frac{d_{\mu,m_0}}{m_0^\alpha}$ thus decreases with m_0, as shown on Figure 5.2-a. It reaches its minimum when all the mass of μ is included in m_0.

In the discrete case, $\delta^2_{\mu,\alpha}$ is the average squared distance to all data points.

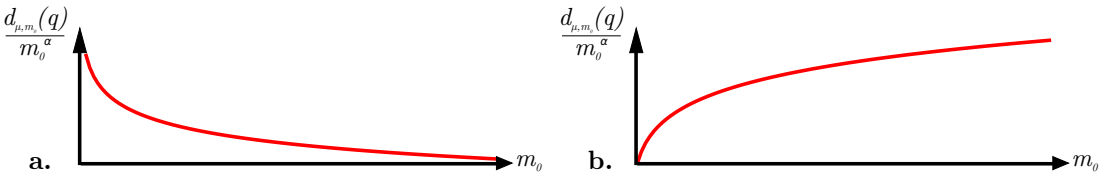


Figure 5.2: **Behavior of $\frac{d_{\mu,m_0}}{m_0^\alpha}$ on ideal cases. a.** Ambient noise in d-dimensional space, $\alpha > 1/d$. **b.** Uniform measure μ on a k-submanifold in d-space, $\alpha < 1/k$.

Submanifold

Consider a uniform measure μ on a k-subspace in a d-dimensional space, as shown on Figure 5.1-b. We denote by $\boldsymbol{p}$ the orthogonal projection of $\boldsymbol{x}$ on the k-subspace. The value of the distance function only depends on the distance to the k-subspace $h = \|\boldsymbol{x} - \boldsymbol{p}\|$ and

m_0 is contained in a hypersphere of the $k-subspace$, of radius r_{μ,m_0} and centered at $\boldsymbol{p}$. A simple calculation shows that:

$$d^2_{\mu,m_0} = \frac{1}{m_0}h^2 \int_{B(\boldsymbol{p},R)} \mathrm{d}\mu(r) + \frac{1}{m_0}\int_0^R r^2 \mathrm{d}\mu(r).$$

The integral of the first term simplifies to m_0, which makes this term equal to h^2. The second term is similar to the calculation in the previous case with dimension k.

$$d^2_{\mu,m_0} = h^2 + \frac{k}{(k+2)(\rho_k C_k)^{2/k}} m_0^{2/k} = h^2 + c_2 m_0^{2/k}.$$

For $\alpha < 1/k$, $d_{\mu,m_0}/m^\alpha$ is unimodal as a function of m_0 as shown on Figure 5.2-b. Its minimum is reached at $m_0 * (h) = c_2 h^k$.

Note that in this case, the noise-adaptive function depends on the distance h from the k-submanifold:

$$\delta_{\mu,\alpha}(h) = c_3 h^{\frac{1}{k}-\alpha}.$$

While the non-adaptive function d_{μ,m_0} is a smooth quadratic function that does not reach 0 on the inferred shape, this modified distance does vanish on the shape and grows quickly, with a vertical tangent, when moving away from it.

The level sets of this adaptive distance function are denser around the input points, providing a more accurate inference of the underlying shape.

Realistic Case

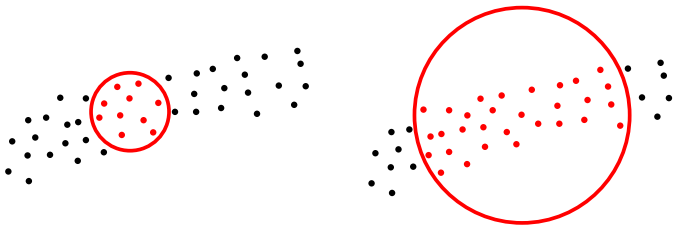


Figure 5.3: **Apparent dimension. Left:** $K = 10$, the apparent dimension is 2 (a surface). **Right:** $K = 30$, the apparent dimension is 1 (a curve).

Consider a k-submanifold sampled by a noisy point cloud. As shown in Figure 5.3, this

can be seen as a mixture of the two ideal cases we just analyzed.

If the scale selected is smaller than the level of noise, then for a query point $\boldsymbol{q}$ located on the inferred shape, the data has the appearance of an ambient noise in d-space as depicted in Figure 5.1-a and $d_{\mu,m_0}/m^\alpha$ decreases the same way as in Figure 5.2-a (provided that $\alpha > 1/d$).

When the scale is large enough to exceed the level of noise, then the data appears as a k-submanifold as depicted in Figure 5.1-b and $d_{\mu,m_0}/m^\alpha$ starts increasing as shown in Figure 5.2-b (provided that $\alpha < 1/k$).

This mixture of both behaviors is described in Figure 5.4.

Under these conditions, $d_{\mu,m_0}/m^\alpha$ reaches its minimum on the value of m_0 that is minimum yet large enough to capture the level of noise, providing a so-called "noise-adaptive distance function". It yields a faithful representation of the inferred shape on noise-free parts while automatically adapting to the local level of noise on areas with poor sampling quality.

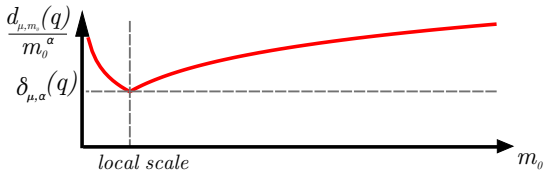


Figure 5.4: **Adaptive distance function.** The minimum of $d_{\mu,m_0}/m^\alpha$ is the noise-adaptive distance function. Its value is reached on a value of m_0 that defines the *local scale*. This curve can be seen as a composition of the two curves from Figure 5.2.

5.1.2 Definitions & Properties

Local Scale & Distance Function

The noise-adaptive distance function is defined this way:

$$\delta_{\mu,\alpha} = \inf_{m_0>0} \frac{d_{\mu,m_0}}{m_0^\alpha}.$$

The parameter α is fixed for a given dimension. More specifically, reconstructing a shape of dimension k in a space of dimension dD implies selecting α in $]\frac{1}{d};\frac{1}{k}[$. In practice,

we set α to the midpoint of the corresponding interval.

- **For curve reconstruction in 2D:** $1/2 < \alpha < 1$, thus we select $\alpha = 3/4$.
- **For surface reconstruction in 3D:** $1/3 < \alpha < 1/2$, so $\alpha = 5/12$.

Note that $\delta_{\mu,\alpha}$ is defined as an infimum over the values of m_0. The value of m_0 corresponding to this infimum is the *local scale* at the considered query point $\boldsymbol{q}$. On discrete cases, this local scale can be referred to as $K(\boldsymbol{q})$ (number K of nearest neighbors needed to get the correct local scale at point $\boldsymbol{q}$).

Figure 5.5 shows a comparison of the non-adaptive distance function to the noise-adaptive one. It also depicts the scale function.

As explained above, this function detects the smallest scale such that the subset of points selected matches the inferred dimension: in this sense, it follows our dimensional *a priori* presented in Section 4.1.

Another noticeable property of this adaptive distance function appears when observing points away from the inferred shape. The different relevant scales of a shape are detected (Figure 5.6).

Like so, points located in the inside part of a 3D torus are local minima of the noise-adaptive distance function: the smallest scale such that the object has the correct dimension at these locations is the scale at which the torus is seen as a single very noisy circle.

This property can be used for the detection of different levels of details or for a multiscale reconstruction. Our work focusing on the reconstruction that preserves as many details as possible above the noise level, we focus on the smallest scale detected that matches the dimension assumption (corresponding to the blue curve in Figure 5.6.

Guarantees

As discussed in Section 4.2.2, the non-adaptive distance function exhibits stability property leading to reliable guarantees for topological inference [23]. We show that the novel adaptive distance function may be modified so as to match similar properties.

For each value of m_0, the distance d_{μ,m_0} is $\frac{1}{\sqrt{m_0}}$-robust: two measures μ and μ' that are ϵ away in the Wasserstein 2-distance sense have robust distances d_{μ,m_0} and d_{μ',m_0} at most $\epsilon/\sqrt{m}$ away in the sup norm.

The square of the non-adaptive distance is also 1-semiconcave. All functions in the infimum are $m_0^{-\alpha-1/2}$-robust and with $m_0^{-2\alpha}$-semiconcave squares.

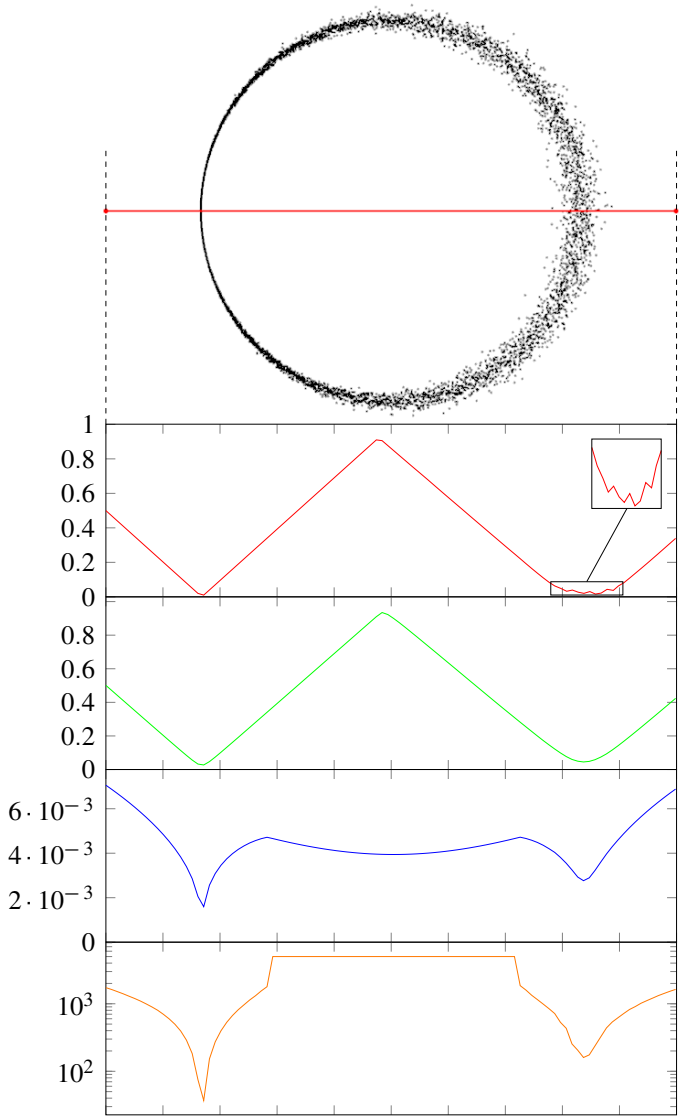


Figure 5.5: **Distance functions. Top:** input point cloud and segment selected to depict function values. **Red curve:** robust function d_μ with $K = 6$: small details are captured in noise-free area, but the function is noisy on noisy area. **Green curve:** robust function d_μ with $K = 70$: noisy areas are captured, but noise-free areas are over-smoothed and the function first minimum has shifted to the right. **Blue curve:** adaptive function δ_μ: all features are captured. **Orange curve:** selected value for K: notice the high dynamic of the function (log vertical scale). The flat maximum appears when the total number of points is reached.

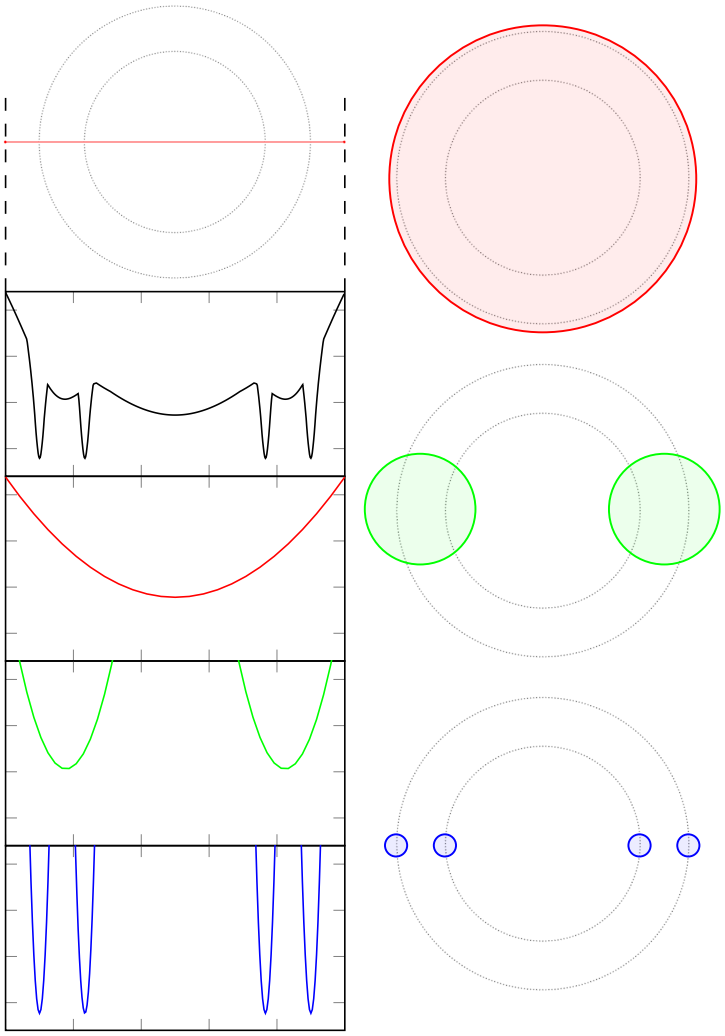


Figure 5.6: **Adaptive function δ_U on a line segment and its scale decomposition. Top:** input point cloud and line segment chosen to depict the function values. **Black curve:** δ_U: all features are captured. The curve may be seen as an infimum of the following curves (corresponding scales shown right). **Red curve:** large scale (K equating the total number of input points). Far from the points, the inferred shape is seen as a point object at the center of mass of the points. **Green curve:** intermediate scale. Between the 2 circles of the ring, the shape is seen as a single noisy circular shape. **Blue curve:** smallest scale. The shape is seen as the noise-free ring sought after.

Since these properties are preserved under taking infimum, so does the modified distance. As robustness weakens on very small values of m_0, we limit the infimum search over values of m_0 that exceeds a lower bound m_{limit}.

In practice, considering discrete measures corresponding to point clouds, we enforce such a lower bound on m_0 by taking at least $K_{limit} = 6$ nearest neighbors.

5.1.3 Computation

Complexity Issue

The time complexity of computing the non-adaptive distance function $d_{\mu,K/n}$ is in $O(K \log(n))$ using a k-d tree, with n the total size of the point cloud and K the number of nearest neighbors (the scale).

Computing exactly the values of $\delta_{\mu,\alpha}$ raises a scalability issue, as it is defined as an infimum search over all possibles values of K between $K_{limit} = 6$ and the number of input points n. For a single query, the time complexity of such a computation using a k-d tree is in $O(n \log(n))$. Massive data sets make such complexity impractical for a large number of distance queries.

Multiscale k-d tree

To alleviate this issue, we approximate the computation of the infimum through a multiscale approach. The main rationale behind this step is that $d_{\mu,K}/K^{\alpha}$ highly varies on small values of K but becomes more and more regular as K grows.

More specifically, the difference between averaging the square distances to the 2 nearest neighbors and to the 3 nearest neighbors is in general larger than the difference between 1002 and 1003 nearest neighbors, for instance.

We compute $d_{\mu,K}$ exactly for small values of K and rely upon far-field approximations for larger values. We proceed as follows:

1. In addition to the original point cloud, we store clustered versions of it obtained through hierarchical clustering [68]. Clusters are parameterized to be uniform, i.e., with no constraints on the local variation (Figure 5.7). We assign to these clusters indices of level of detail i, with index 0 for the input point cloud. Denote by s the

size of each cluster (set by default to 10 in all shown experiments). The size of the i^{th} clustered version of the point cloud is $\frac{n}{s^i}$.

2. We build a k-d tree for each of these point clouds, including the unaltered input point cloud.

3. To evaluate the function $d_{\mu,K}/K^{\alpha}$, we use in sequence these k-d trees with increasing K until reaching the desired K. More specifically, we approximate the distance $d_{\mu,K}$ by increasing K and summing up squared distances to cluster points with appropriate weights s^i. When K gets larger than s^i, we switch to the next k-d tree with index $i+1$.

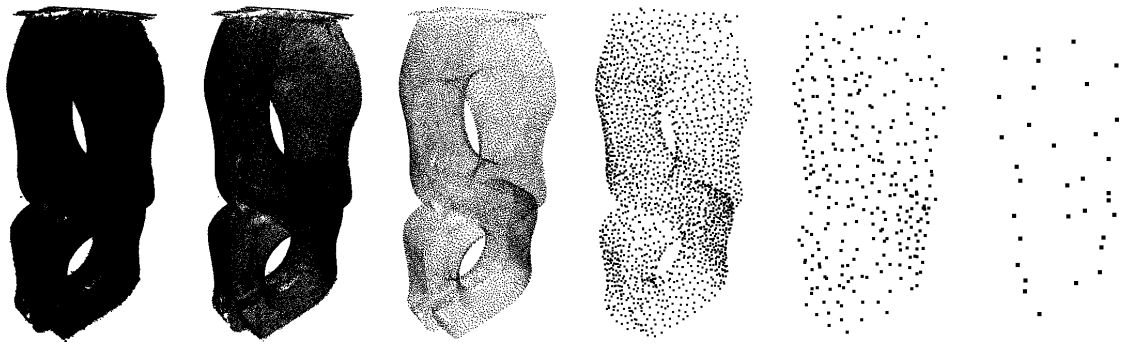


Figure 5.7: **Multiscale k-d tree.** The input point cloud (1M points) is hierarchically clustered with clusters of size 10.

The time complexity for a single query drops to $O(\log(n)^2)$, which solves the scalability issue while providing satisfactory approximations of $\delta_{\mu,\alpha}$ as shown by Figure 5.8.

5.1.4 Representation

For fast queries on $\delta_{\mu,\alpha}$, we wish to compute it on an adapted representation of the domain and to approximate it through piecewise-linear interpolation.

We compute a triangulation (tetrahedralization in 3D) through Delaunay refinement. For scalability purpose, this triangulation is made adaptive, with smaller cells on areas where the function gradient varies rapidly.

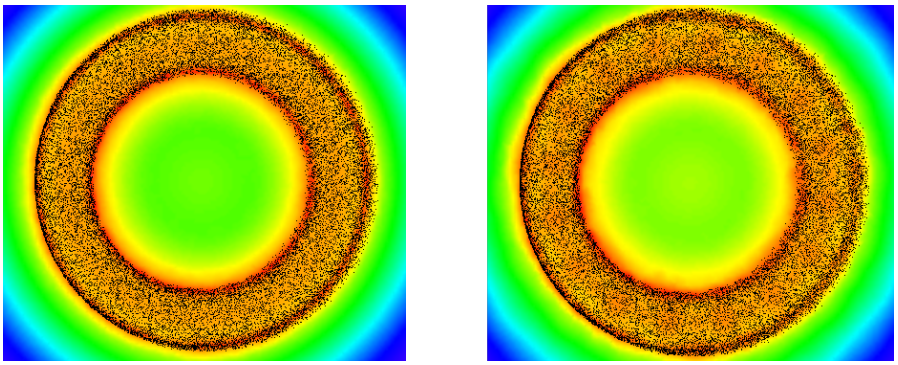


Figure 5.8: **Approximating δ_μ.** The 3D point cloud (50K points) is sampling a torus with variable noise. The functions are depicted on a planar slice. Left: exact computation (3 minutes). Right: approximation with a multi-scale k-d tree (1 second).

Delaunay Refinement

We refine a Delaunay tetrahedralization via Delaunay refinement [71]: starting with a loose convex hull of the input point cloud, the tetrahedralization is greedily refined by inserting Steiner vertices until the cells fulfill a shape criterion. The latter is based on the radius-edge ratio, that is to say the circumradius of a cell divided by its shortest edge. A cell is considered a bad tetrahedron if its radius-edge ratio is larger than a given threshold. In this case, the circumcenter of this tetrahedron is added as a vertex of the triangulation.

Error Probing

After refinement, the triangulation must provide a faithful piecewise linear interpolation of the distance function. We denote by $\tilde{\delta}_{\mu,\alpha}(\boldsymbol{q})$ the interpolated value of $\delta_{\mu,\alpha}(\boldsymbol{q})$. We define an interpolation error at point $\boldsymbol{q} \in \mathbb{R}^3$:

$$e(\boldsymbol{q}) = \|\tilde{\delta}_{\mu,\alpha}(\boldsymbol{q}) - \delta_{\mu,\alpha}(\boldsymbol{q})\|.$$

The refinement is guided by the error, keeping it under an error criterion. In addition to checking the radius-edge ratio of a cell, we also pick a number p of random points on the cell, the so-called *probes*. On each of these probes, we test the value of the error $e(\boldsymbol{q})$: as soon as one probe gives an error above a threshold e_{max}, the cell is considered as bad and we insert its Steiner vertex. If a user-defined maximum number of probes p_{max} return

an error lower than e_{max}, the cell is considered good and the refinement stops (provided the radius-edge ratio criterion is also met).

Such refinement algorithm generates tetrahedralizations with a high density of vertices in areas where the function $\delta_{\mu,\alpha}$ varies quickly and lower density elsewhere (see Figure 5.9).

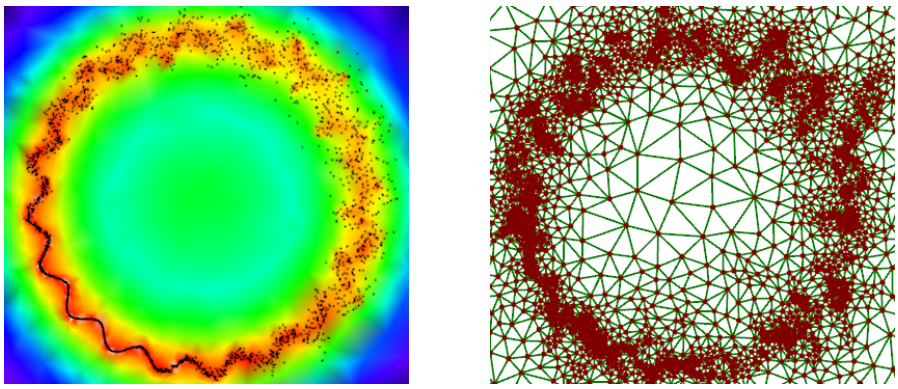


Figure 5.9: **Delaunay refinement.** Left: input point cloud and distance function. Right: tetrahedralization computed with $p_{max} = 15$ and $e_{max} = 0.001$.

5.2 Guessing the Sign

We now wish to compute a signed implicit function from the noise-adaptive distance function. We first guess the sign on a discrete set of points covering the scene. Our approach is inspired by the ray shooting and parity test step performed by Mullen et al. [64] with some differences an improvements.

Because of the high variations of the noise-adaptive distance close to the inferred shape, we first want to guess the sign away from the input points, where confidence is higher.

5.2.1 Segment Picking

Limitation of Ray Shooting

One of the major problems of ray shooting arises from rays near tangent to the inferred shape, as depicted by Figure 4.3. As the shape gets wider and more complex, the probability of a wrongly estimated parity increases.

Note that when a parity test fails, it only means that the ray is aligned with the inferred shape at some point or that it crosses a region with missing data. Other parts of the ray might as well be valid, but as the parity test uses the whole ray, the whole ray becomes useless.

Using Segments

Ray shooting may be seen as a very specific case of a more general process referred to as "segment picking". Indeed, when representing a scene on a computer, shooting a ray is equivalent to picking a segment whose end points are the query point and a point at infinity.

We generalize this idea by picking segments with end points sampled everywhere on the domain. We use both small segments which have little probability of being tangent to the shape, and long segments which provide a means to improve robustness to missing data.

Our approach randomly selects a large number of segments within the scene, the number of segments selected being our means to trade robustness for computational time.

5.2.2 Signing a Segment

Performing a parity test on a segment consists in estimating the number of times it crosses the inferred shape. Mullen et al. [64] use the precomputed band where $d_{\mu,K} < \varepsilon$.

Although it also drops near the inferred shape, the noise-adaptive function $\delta_{\mu,\alpha}$ is not suitable to such a constant thresholding: the range of values it takes near the inferred shape depends on the local scale and point density, see Figure 5.5 (blue curve).

Smoothness Assumption

The noise-adaptive distance function drops near the inferred shape. These areas are in general sharper than the other local minima found on a segment. Our rationale is that the signed implicit function should be smoother after switching signs near the inferred shape.

Relying solely upon a smoothness assumption, we proceed as follows: among all possible ways to sign the distance function on a segment, we select the one that generates the smoothest signed function.

First, we compute the interpolated approximation of the function $\tilde{\delta}_{\mu,\alpha}$ along the segment and extract all the local minima. To reduce the number of local minima that can arise on noisy areas, we first perform a light Gaussian filtering of this function.

As the function does not reach 0 on the inferred shape, we replace sign flips by *local flips*: the function is mirrored with respect to the horizontal line located at each local minimum for each segment. The local sign guess problem is formulated as a combinatorial problem of complexity 2^N, where N denotes the number of local minima.

We test all possible combinations of local flips. To evaluate the smoothness of each solution, we use the squared norm of its second derivative, measured and summed up at multiple scales and computed through finite differences after uniform discretization of $\tilde{\delta}_{\mu,\alpha}$ along the segment.

Figure 5.10 depicts the signing process for one segment.

5.2.3 Signing a Graph

From the signing process described above for one segment, we can perform both local and non-local sign hypotheses via randomly picking short and long segments. We next wish to consolidate these hypotheses via a global solve for signing the vertices of a graph.

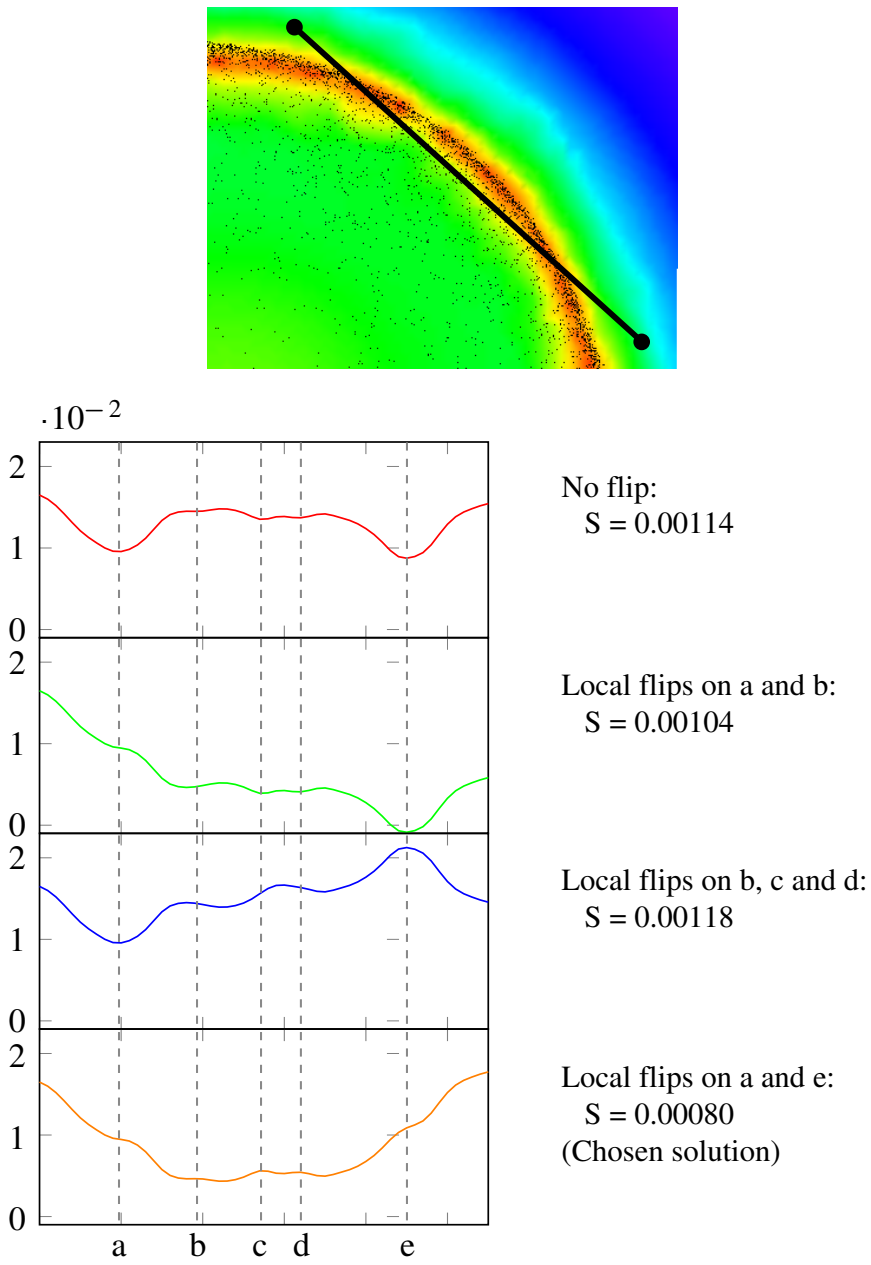


Figure 5.10: **Signing through local flips at local minima.** Top: 5 minima found on the segment. Top curve (red): no flip and corresponding smoothness value (S). Bottom curve (orange): the chosen combination of local flips corresponds to the smoothest curve.

In the ray shooting approach, one of the end points of a ray is at infinity and therefore known to be outside. The sign of the other end point can thus be directly determined by the parity of the ray. This is not the case when using finite segments: a single sign hypothesis provides only a means to estimate if the end vertices of an edge have similar or opposite signs.

To determine the signs of end points from segment hypothesis, we propose a novel approach based on a random graph.

Definition

We generate a random graph G as follows:

- Nodes are located on the vertices of a regular grid that covers the scene.
- Edges are created by randomly picking a large number of pairs of nodes. Using the parity test previously described, each of them is assigned a sign hypothesis -1 when the two end nodes are estimated to have the same sign, $+1$ otherwise.

The user must specify two parameters: a number of nodes and a number of edges randomly picked. Large numbers provide a more reliable sign guess at the cost of longer computation times.

Note that we also experimented with a non-regular graph with nodes placed on the vertices of the adapted tetrahedralization. Many nodes are located near the inferred shape which makes the parity test fail in most cases. The failures do not disappear even when compensating for the variable density of nodes with appropriate weights.

Energy Minimization

The nodes are given indices i. An edge is denoted by the indices of its end points (i, j). s_i denotes the unknown sign guess function at node i and $\varepsilon_{i,j} = \pm 1$ denotes the sign hypothesis on the segment (i, j).

On the graph G, we define and minimize the following energy:

$$E_G(f) = \sum_{(i,j)\in G} (s_i + \varepsilon_{i,j} s_j)^2.$$

The goal of this minimization is to solve for a global consensus on the sign of nodes based on the sign hypotheses found at each segments. To avoid the trivial solution where f is 0 everywhere, we constrain the solution to have average value equal to 1: $\frac{1}{|G|}\sum_{i \in G} s_i = 1$.

Note that we could also constrain the nodes on the boundary of the scene to be 1, relating to ray shooting where the points away from the shapes are assumed to be outside of the inferred shape. Nevertheless, this is a poor constraint if a shape is partially scanned. Another solution is imposing the solution to have unit L^2 norm. However, this requires solving for an eigenvalue problem with limited scalability.

The energy E_G is minimized using a sparse linear solve: we use a conjugate gradient algorithm applied to a sparse matrix, using a *conjugate gradient* algorithm provided by the Eigen library [42]. Solving for a graph with 300K nodes and 20M edges takes about 10 seconds in our experiments (see Section 5.4). The random edge selections with sign hypotheses tests are trivially parallelized.

A random graph is depicted by Figure 5.11.

Confidence on Solution

As some nodes may randomly lie on the inferred shape or on a part with missing data, we identify the most confident nodes.

The confidence c_i of a node i is defined as a ratio of match hypotheses:

$$c(i) = \frac{1}{N}\sum_{j=0}^{N} S(i,j),$$

where N denotes the number of edges adjacent to i and $S(i,j) = 1$ if s_i and $\varepsilon i, j \times s_j$ have opposite signs, 0 otherwise. Figure 5.12 depicts three different confidences.

All nodes with confidence higher than a user-specified threshold ($c > c_{min}$ with $c_{min} = 0.75$) are referred to as *confident nodes* and used in the next section. Figure 5.13 depicts such a thresholding.

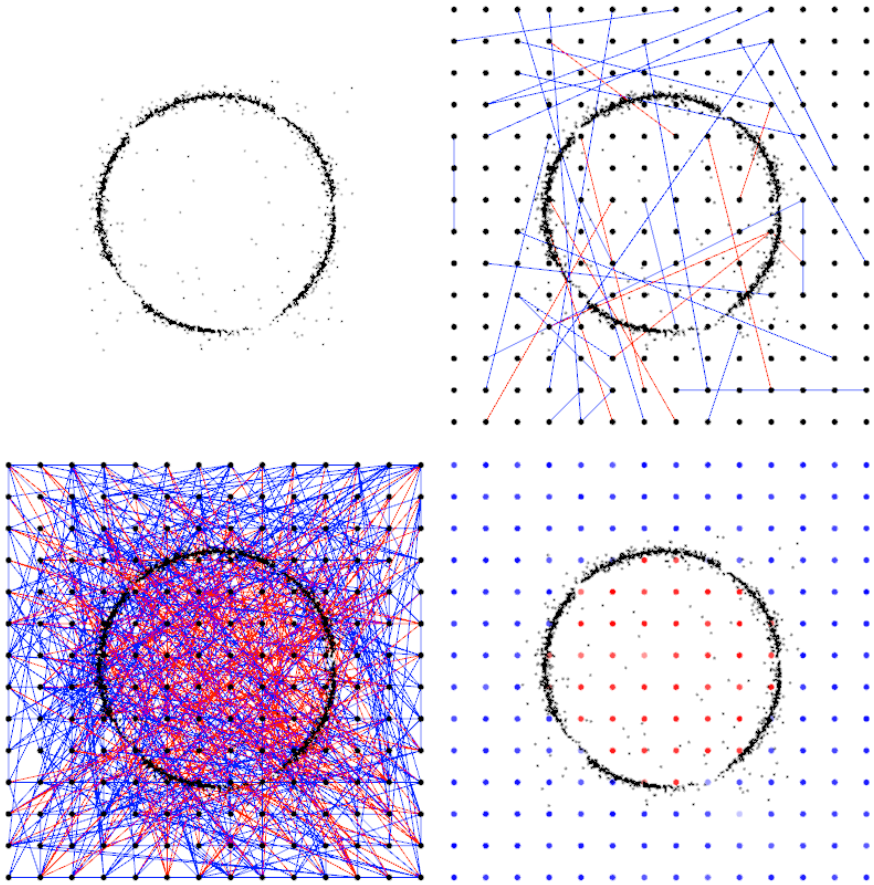


Figure 5.11: **Random graph.** Top: input point cloud and edges of the graph (only 1% of edges are shown for clarity, with blue for similar signs and red edges for different signs). Bottom: 20% of the graph edges shown, and signed function at graph nodes after linear solve (red for inside, blue for outside).

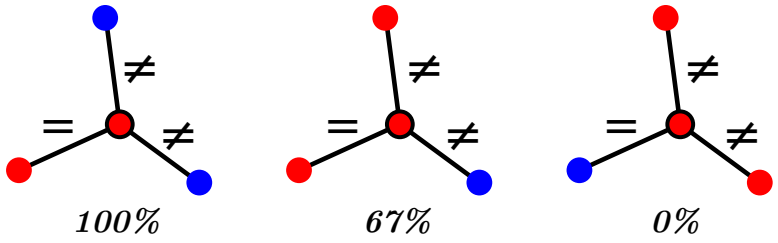


Figure 5.12: **Confidence of a node.** The signs of the nodes are depicted in red and blue. The hypotheses are written with the symbols = and ≠.

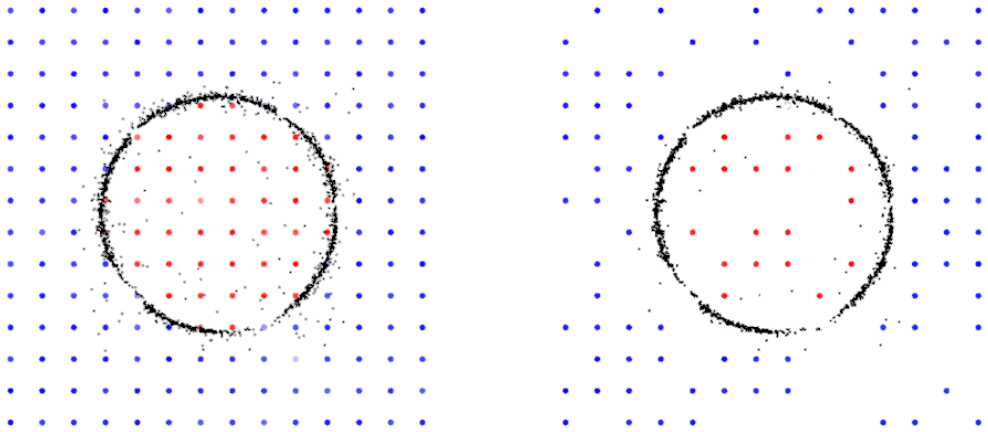


Figure 5.13: **Confidence filtering.** Left: result of the sign guess. Right: sign guesses with a confidence $\geq c_{min} = 0.75$.

5.3 Signing the Distance Function

We now detail the final step of the algorithm: converting the unsigned noise-adaptive distance function to the signed implicit function using the pointwise sign guesses computing in previous step (Figure 5.14).

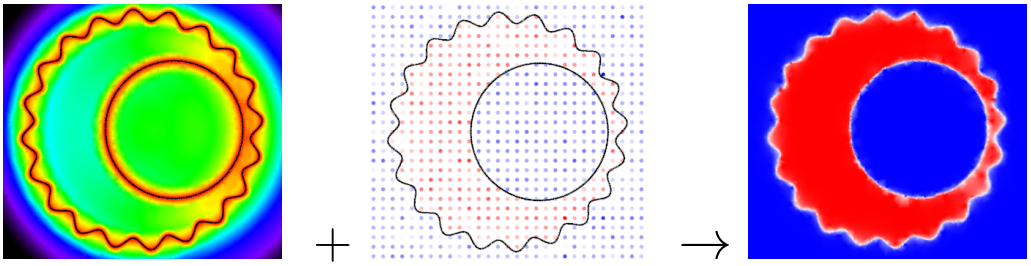


Figure 5.14: **Signing the distance function.** From left to right: input distance function, confident sign guesses and output implicit function.

5.3.1 Intuition

Given the set of most confident signed nodes computed in the previous step, we wish to compute a global signed function whose 0-isolevel is a reliable reconstruction of the inferred shape.

We denote by g the signed implicit function. It is defined on the adaptive tetrahedralization T as for $\delta_{\mu,\alpha}$.

Intuitively, we want the signed function to fulfill two constraints:

- Match the most confident sign guesses.
- Switch sign on the low values of $\delta_{\mu,\alpha}$.

This idea is depicted on Figure 5.15. The rationale is that the inferred shape is located in areas where $\delta_{\mu,\alpha}$ takes low values and where the sign is estimated to change.

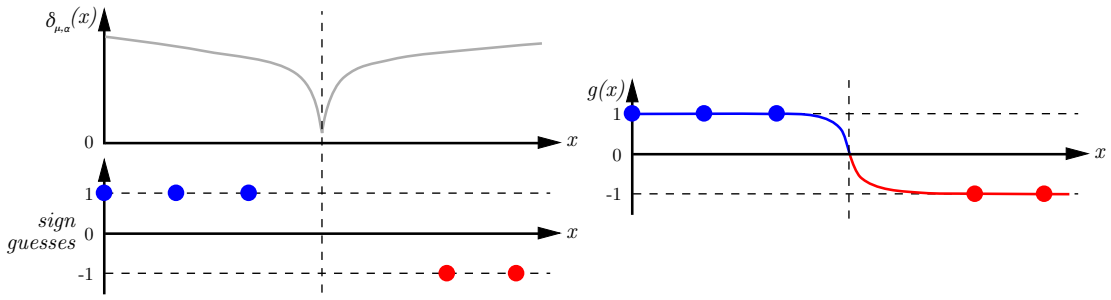


Figure 5.15: **Signing step, input and output.** Left: input distance function and sign guesses. Right: output signed implicit function.

5.3.2 Energy Minimization

To compute the signed function, we propose a method inspired by the *random walker* approach used for image segmentation [40]. It is primarily used to label specific parts of an image, by identifying borders defined by pixels with high gradient value.

In our context, we define only two labels (*inside* and *outside*) and a set of seeds (the signed nodes). We define a cost function:

$$w(x) = \begin{cases} \delta_{\mu,\alpha}(x) & \text{if } K(x) < K_{max} \\ +\infty & \text{otherwise} \end{cases}$$

We then search for a function g that minimizes the weighted Dirichlet energy:

$$E_{g,T} = \int_{\Omega} w(x)|\nabla g(x)|^2 \mathrm{d}x.$$

The rationale is to compute an approximate indicator function of the inferred shape where the function varies abruptly. The upper bound on the local scale K_{max} prevents the sign from switching inside thin features that can have a low local scale while irrelevant for reconstruction. We set by default $K_{max} = 500$.

A trivial constant solution would be found when no other constraints are defined: the reliable sign guesses computed in previous step are used to avoid it. As the energy is computed on the adapted tetrahedralization, we simply assign sign guesses of the confident nodes to their closest vertices in T. These guesses are used as soft constraints by the minimization step.

Solving for this energy in the space of piecewise-linear functions defined on T boils down to solving the linear system $(L + \alpha C)X = \alpha B$, where:

- L is a weighted Laplacian matrix of size $V \times V$, V being the number of vertices of T.
- α is a user-specified coefficient used to weight the influence of constraints.
- C is a diagonal matrix with $c_{i,i} = 1$ if vertex i is constrained, 0 otherwise.
- X is the solution vector solved for.
- B is the right hand side vector where $b_i = 0$ if the vertex i is unconstrained, and $b_i = \pm 1$ depending on the sign constraint otherwise.

Parameter α is set empirically in the range of $10^{-3} \times \frac{\text{trace}(L)}{v}$.

Figure 5.16 summarizes the signing pipeline ranging from the noise-adaptive distance function to the implicit function.

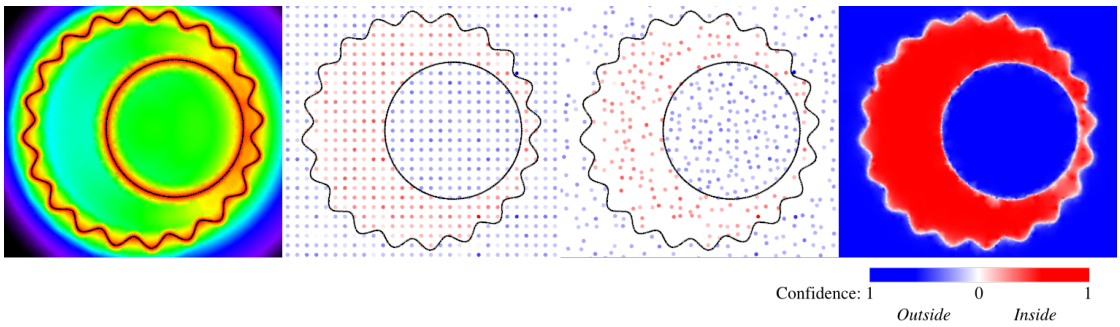


Figure 5.16: **Signed implicit function.** Top: unsigned function and sign guess on nodes of uniform graph. Bottom: confident nodes used as soft constraints and signed implicit function after linear solve.

5.4 Results

The algorithm is implemented in C++ using the *CGAL library* [1] and the *Eigen library* [42] for least squares fitting. All experiments are performed on an *Intel* 2.4GHz laptop with 8GB RAM.

Although the output of the algorithm is an implicit representation of the reconstructed shape, we extract an isosurface by using the isolevel corresponding to the median value of $g(x)$ on the input points:

- In 2D, the curves are obtained through marching triangles on the 2D adaptive triangulation.
- In 3D, the surface are obtained through Delaunay refinement [71]. Marching tetrahedra are another solution but generates overly complex meshes.

Timings and parameters are provided by Figure 5.17.

Point cloud	**Size**	$\mathbf{T_{dist}}$	$\mathbf{T_{guess}}$	$\mathbf{T_{reco}}$	$\mathbf{T_{total}}$	**Memory**	**Specific parameters**
Figure 5.18	857.726	207	32	12	242	700 MB	
Figure 5.21	50.000	52	25	4	82	300 MB	
Figure 5.24	50.000	91	33	6	130	400 MB	$K_{max} = 1000$
Figure 5.25	419.488	181	155	6	342	1.6 GB	$e_{max} = 0.003$, $500K$ nodes, $15M$ edges, $l = 0.60$

Figure 5.17: **Timings.** Timings in *seconds*. Default parameters: $p_{max} = 10$, $e_{max} = 0.004$, $50K$ nodes, $1.5M$ edges, $c_{min} = 0.75$, $K_{max} = 500$.

5.4.1 Perfect Data

Albeit our algorithm is designed to deal with variable noise, we first experiment with defect-free point clouds.

Figure 5.18 shows a reconstruction of the *genus* point cloud. It is close to noise-free with few holes due to missing data. It was generated by photogrammetry (courtesy *EPFL*

Computer Graphics and Geometry Laboratory [2]). We compare our reconstruction to the Poisson surface reconstruction [48]. Normal vectors provided with the original point cloud are used by the Poisson reconstruction step.

Figure 5.18: **Low noise.** From left to right: raw point cloud; point cloud & reconstruction; reconstruction only; Poisson reconstruction.

Our algorithm yields comparable results with some noticeable differences on missing data.

5.4.2 Uniform Noise and Outliers

The primary added value of our approach is its robustness to both uniform noise and outliers. Figure 5.19 depicts the stability of our approach against increasing amounts of noise and outliers.

Note that a correct reconstruction is still obtained even when the estimation of oriented normals is impossible. We also show some failure cases when the combined effects of noise and high density of outliers challenges our local dimension assumption.

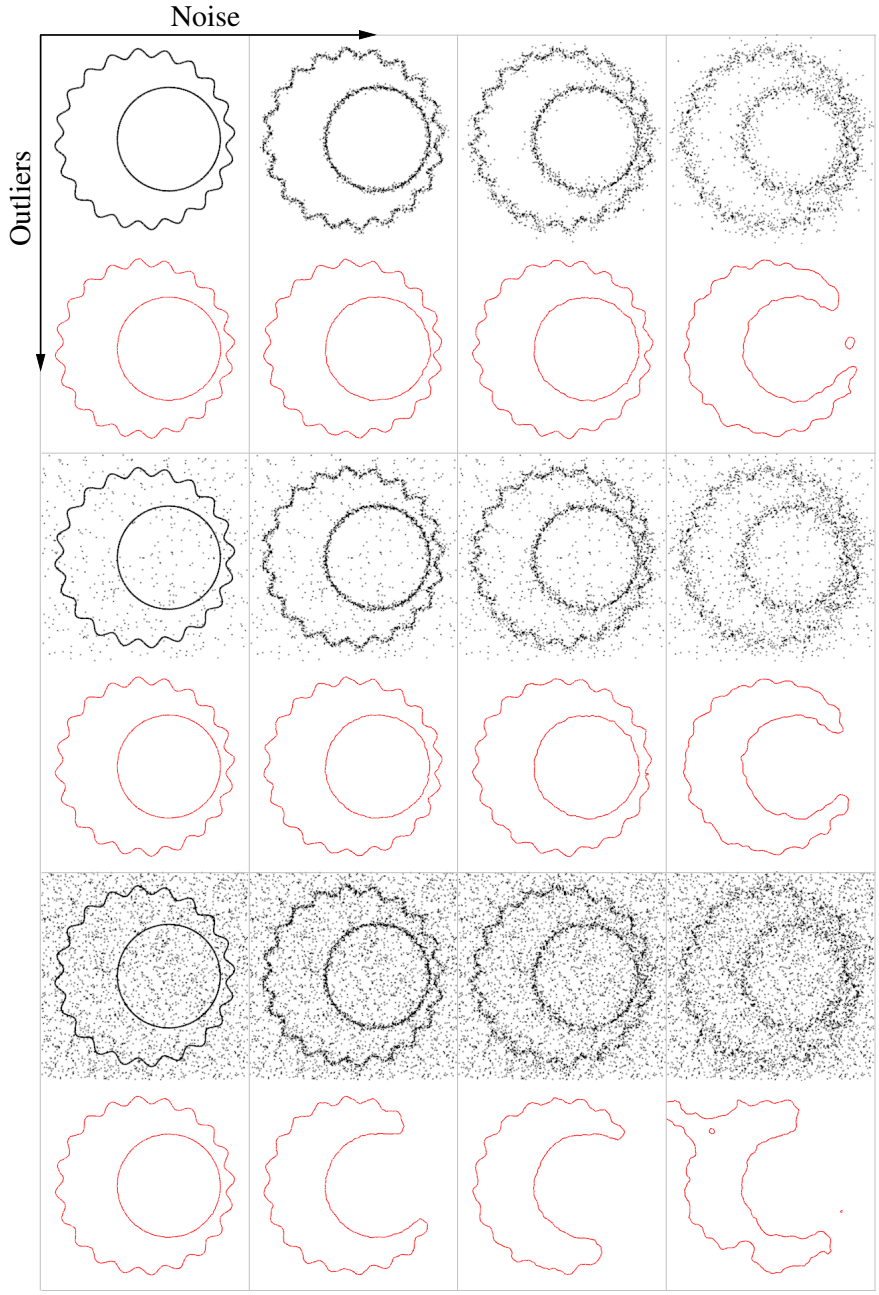


Figure 5.19: **Noise and outlier robustness.** The noise increases from left to right. The outliers increase from top to bottom, ranging from outlier-free to 60% through 20%. Input point cloud in black, reconstructed curve in red.

5.4.3 Variable Noise

Our algorithm outperforms on automatic and adaptive detection of local scales. As shown by Figure 5.20, a constant scale parameter K fails in reconstructing shapes with variable levels of noise: noise-free areas are oversmoothed when K is selected large enough to be robust to the noisy area.

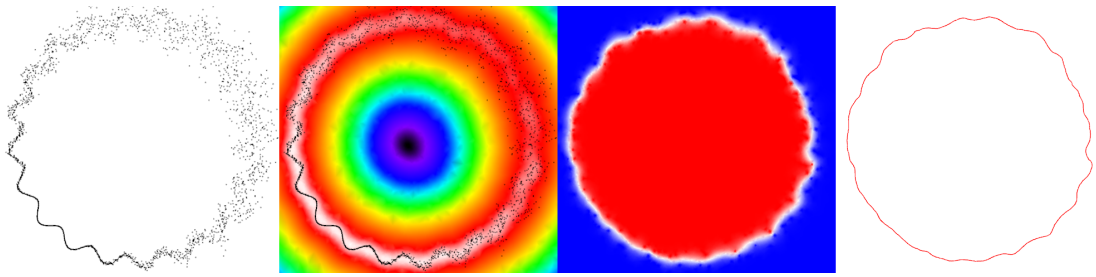


Figure 5.20: **Reconstruction with a constant scale ($K = 80$).** From left to right: input point cloud with variable noise; unsigned distance function to the inferred shape; sign guess and confidence; reconstructed shape, over-smoothed.

To challenge our algorithm, we generate a variety of variable noises on several point clouds.

Gradually Variable Noise

We first generate a variable noise by increasing linearly the noise level.

Figure 5.21 illustrates a relevant case where small features in noise-free areas have similar scale as the noise in difficult areas. Our approach smoothly approximates the inferred shape on noisy area while providing high accuracy on noise-free areas.

The Poisson reconstruction, on the contrary, fails at dealing with variable noise: either the scale is large enough to handle noise, and the detailed features are lost or the scale is small enough to reconstruct the detailed features but gives poor results in noisy areas with incorrect topology.

Another strength of our method is its capability to reduce shrinkage in curvy areas (Figure 5.22).

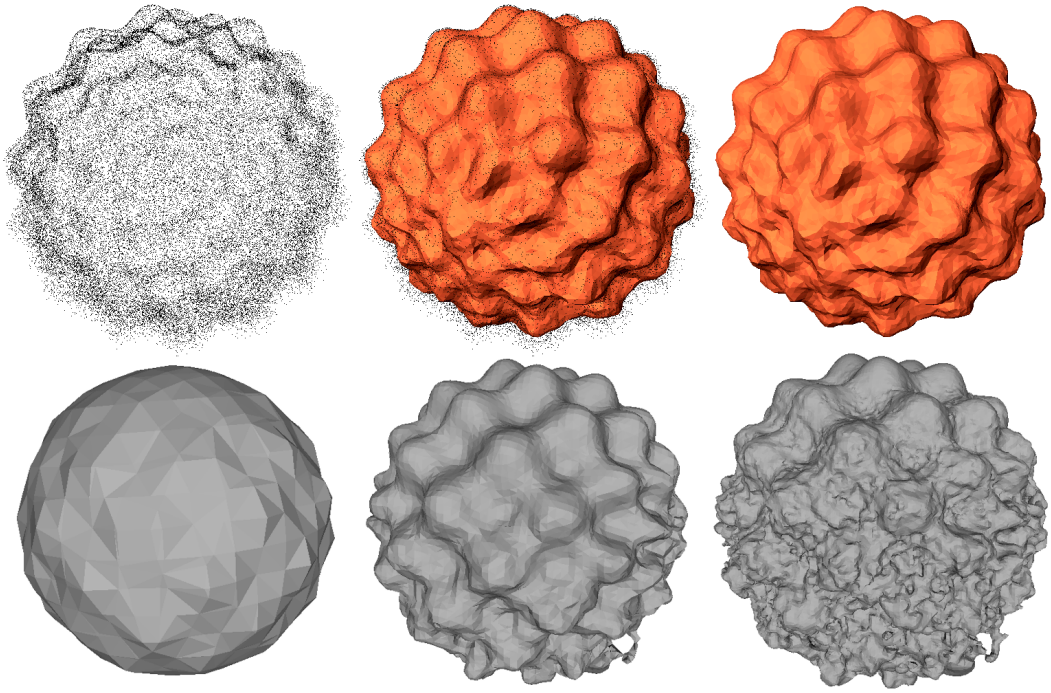


Figure 5.21: **Gradually variable noise (generated).** Top: raw point cloud, where noise increases linearly from top to bottom; point cloud & our reconstruction; our reconstruction only. Bottom: Poisson reconstruction with a constant octree depth of 4, 6 and 8.

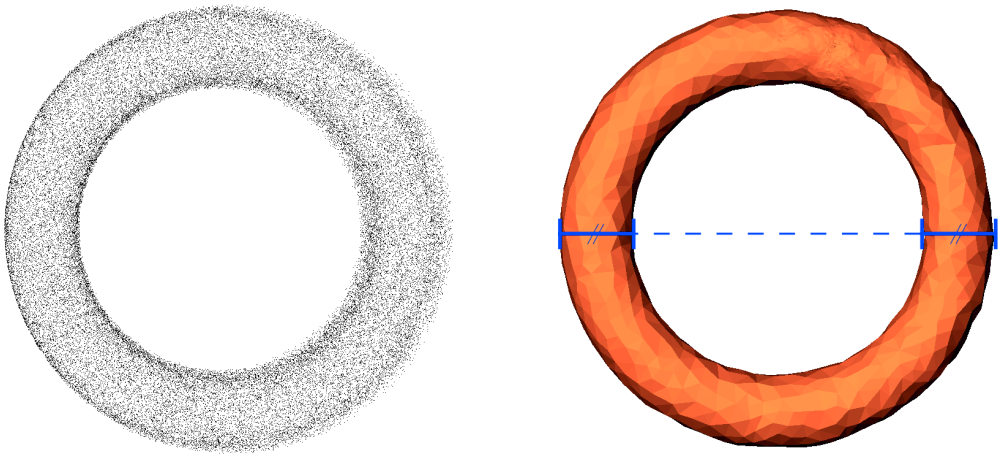


Figure 5.22: **Variable noise (generated).** Left: input point cloud. Right: reconstructed surface. Notice that the reconstructed surface exhibits no shrinkage.

Abruptly Variable Noise

We now experiment with another type of variable noise, with a noise-free area clearly separated from the noisy area. Such type of noise appear when heterogeneous devices or acquisition conditions are used.

Half of the genus model is tampered by noise (Figure 5.23). Our algorithm yields a valid reconstruction with a smooth transition at the interface between the two noise levels. The noise-free part is not affected by this change compared to Figure 5.18.

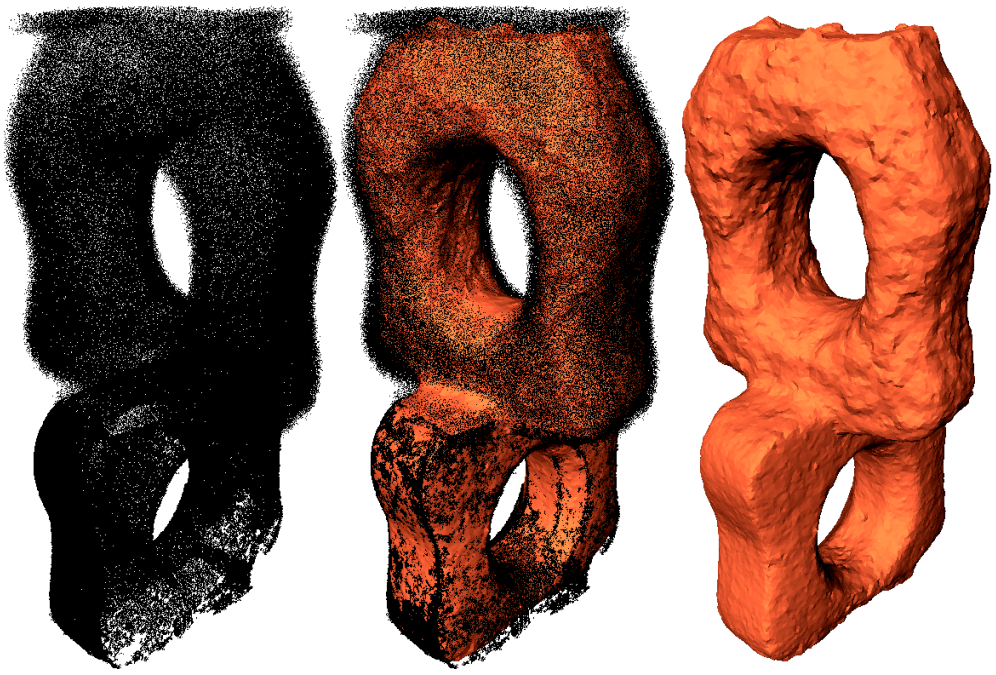


Figure 5.23: **Two levels of noise.** Left: raw point cloud with additional noise on the top half part. Middle: point cloud & reconstruction. Right: reconstruction only.

We furthermore generate a point cloud with two interlaced tori hampered by two levels of noise (see Figure 5.24).

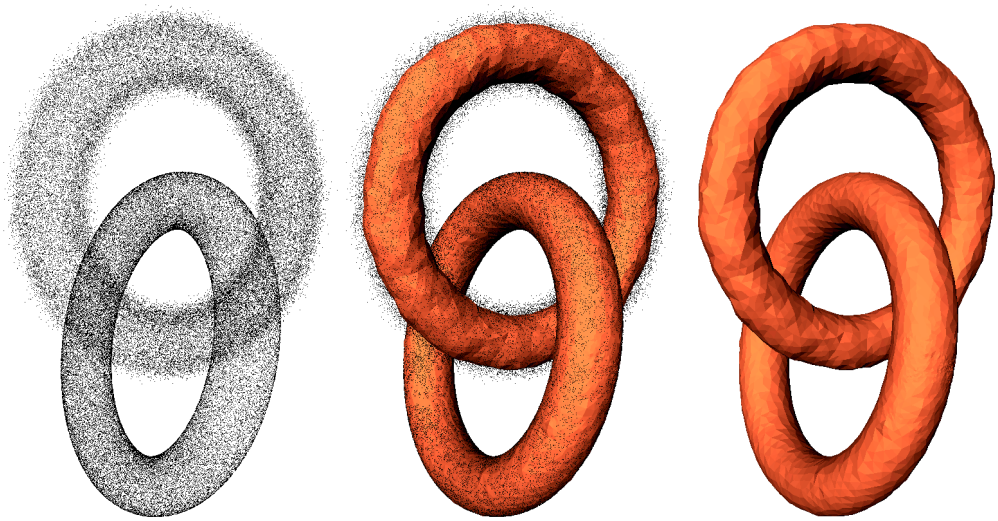


Figure 5.24: **Two levels of noise (generated).** Left: raw point cloud containing a noise-free and a noisy torus. Middle: point cloud & reconstruction. Right: reconstruction only.

5.4.4 Structured Outliers

The *capitol* model (Figure 5.25) challenges our algorithm. It was acquired through dense photogrammetry and contains some artifacts in the form of structured outliers as well as variable noise. All normal estimators fail on this point cloud.

Our algorithm reconstructs the inferred column capitol. While most of the details are lost, it successfully locates the inside of the shape and is not fooled by the "ghost shape" induced by structured outliers.

5.4.5 Failure Cases

We push our algorithm to its limits, up to failure cases.

Excessive Noise

When the noise level is extreme, the dimension assumption may not be valid anymore, see Figure 5.26. The dimension of the underlying shape becomes ambiguous and our reconstruction algorithm fails in producing a reliable reconstruction. Note that the correct

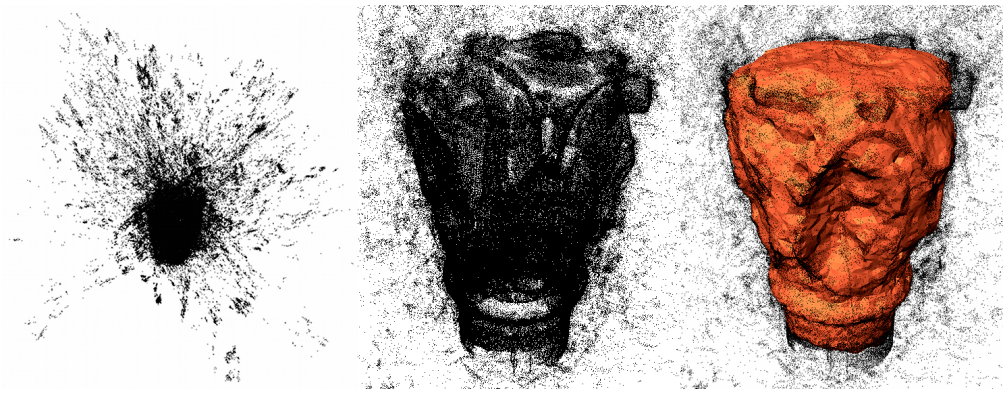


Figure 5.25: **Noise and structured outliers.** Left: raw point cloud. Middle: closeup on point cloud. Right: closeup on point cloud & reconstruction.

topology is still captured even while the geometry of the object is poorly reconstructed.

Figure 5.27 illustrates a failure case with an extreme level of noise: the noisy part of the torus has the same apparent dimension than a noisy circle embedded in 3D. The reconstruction degrades rapidly on such areas.

Non-Uniform Sampling Density

We conjecture that robustness to outliers is not compatible with robustness to widely variable sampling, as a very low sampled area can not be distinguished from a set of outliers.

Figure 5.28 illustrates our approach at work on such a point cloud.

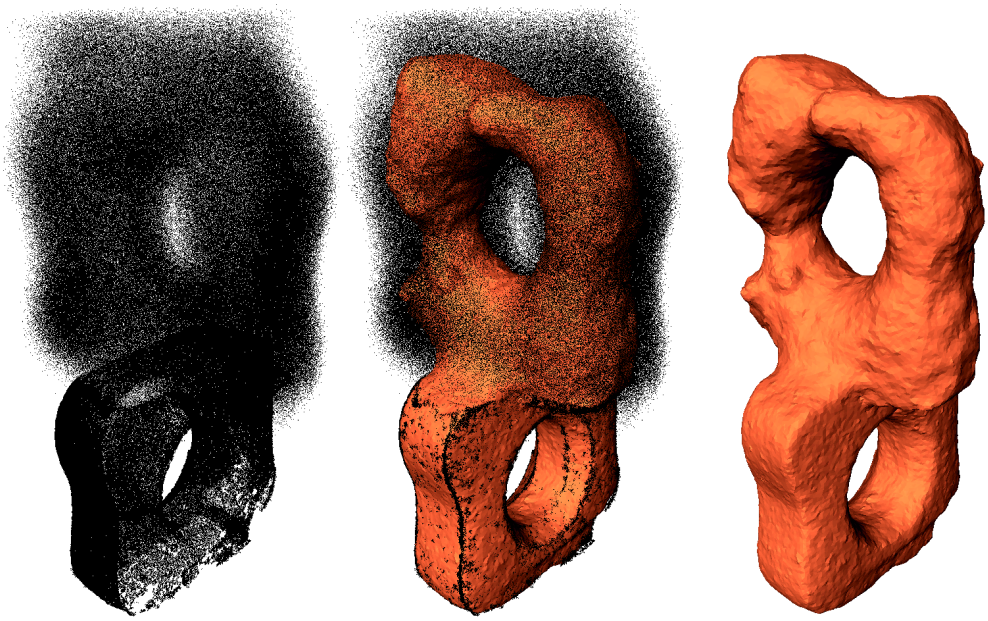


Figure 5.26: **Noise almost beyond dimension assumption.** Left: raw point cloud with high noise on the top half part. Middle: point cloud & reconstruction. Right: reconstruction only.

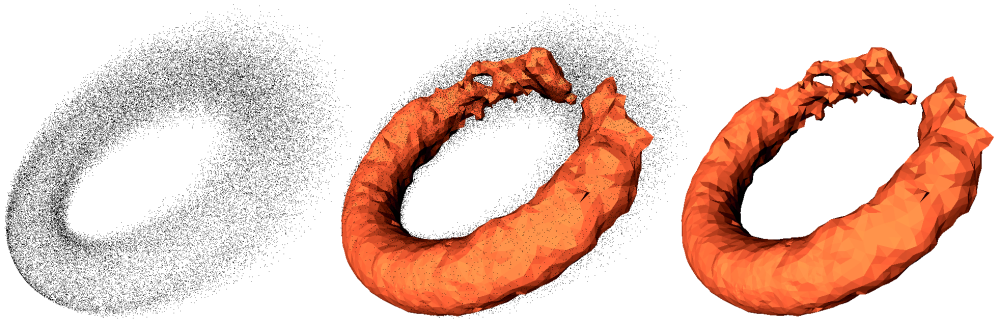


Figure 5.27: **Noise beyond dimension assumption (generated).** Left: raw point cloud. Middle: point cloud & reconstruction. Right: reconstruction only.

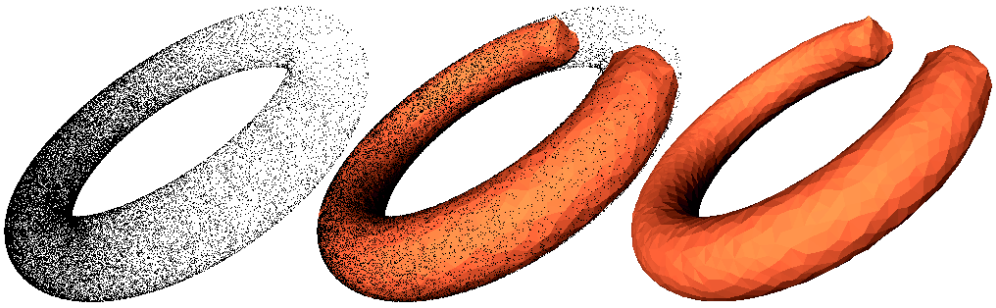


Figure 5.28: **Variable density (generated).** Left: raw point cloud. Middle: point cloud & reconstruction. Right: reconstruction only. Our method fails in capturing the correct dimension in the low density area.

5.5 Conclusions

5.5.1 Technical Contribution

We have introduces a novel shape reconstruction algorithm for smooth closed shapes. Our main contribution is a novel distance function that automatically and locally adapts to variable levels of noise. The novel noise-adaptive distance function stems from a robust distance function to a measure that provides resilience to noise and outliers. It involves only two linear solves and relies solely on a dimension assumption.

Our approach handles some non-trivial point clouds that are not handled by other methods, or poorly. Robustness to variable noise is achieved by assuming that the inferred shape is a smooth submanifold of known dimension.

Our second contribution is a novel approach for guessing the sign of a finite set of points. By extending the concept of ray shooting to segment picking, we increase the robustness of the signing step on complex shapes with large amount of missing data.

The algorithm only involves solving two linear systems on sparse matrices, which scales to large data sets.

Finally, the noise-adaptive distance function is sensitive to the different scales of a single shape. In future work, we plan to devise an automated approach to select the intrinsic scales for geometry processing or to perform multiscale or hierarchical shape reconstruction.

5.5.2 Limitations

The main limitation of our reconstruction pipeline is the need for a regular graph: this hampers scalability in the presence of very many details. As the distance between two nodes must be smaller than the smallest detail, this can yield a very large number of nodes to cover the whole scene. It remains to find a principled way to generate a non-uniform graph.

Another limitation lies in the parity test of an edge of the graph: the total number of combinations (2^N possible flips) increase rapidly with the amount of details in the scene. In our experiment, this number is around 6 on average, but we can imagine a complex shape with many surface sheets that would yield many more local minima.

Finally, our signing approach proceeds in two steps: estimating the sign on a discrete set of points and then solving for the implicit function. It would be more elegant to perform the whole signing of the distance function in one single step without hampering the scalability.

II

Piecewise-Planar Reconstruction

6 State of the Art

6.1 Introduction

Piecewise-planar reconstruction greatly differs from reconstruction of smooth closed shapes. Although some methods presented in Chapter 2 can to some extended handle piecewise-smooth surfaces, dealing with opened surfaces composed solely of planar sections has been explored independently, for example in urban modeling from multi-view or laser data.

The scientific challenges induced by piecewise-planar reconstruction also differ. Reconstruction of smooth closed shapes relies on strong assumptions on the output shape: noise robustness can be achieved through smoothness assumptions while non-uniform sampling or missing data are taken into account in the view of reconstructing a closed surface.

For piecewise-planar reconstruction, the assumptions on the output shape are relaxed. The shape may include sharp features and is not bound to enclose an inside volume. To some extend, smooth parts can also be approximated by tangential planar primitives.

The problem of reconstructing a surface with planar primitives can be tackled by several approaches: planar parts need to be identified and their boundaries well defined. An adjacency graph between the different planar parts should also be found in order to properly embed the output mesh.

Depending on whether one of these steps is performed first and whether it is done

one way or another, the reconstruction methods differ. In this chapter, we first review the methods based on *primitive detection*. We then review some *feature*-based techniques and briefly review some *variational* approaches.

6.2 Primitive-based reconstruction

Piecewise-planar reconstruction can be represented as a set of planar surface parts, along with their boundaries and relative adjacencies. A first approach to produce a piecewise-planar reconstruction from a point cloud is to first build planar primitives, then deduce their boundaries and relative adjacencies afterwards. Robustness to noise and missing data is also sought after.

RANSAC

The literature in shape detection is vast. The output differs in the sense that it does not represent the shape as a global object (mesh, implicit function, etc.) but as a disconnected set of primitives.

Random Sample Consensus [35] (referred as RANSAC) is a popular shape detection approach that handles different types of primitives: planes, but also canonical primitives such as spheres and cylinders. In this review, we focus on plane detection. The original RANSAC algorithm repeats the following process:

1. Select a random subset of the input point cloud (*hypothetical inliers*).
2. Fit a plane to these inliers.
3. Find all inliers of the plane from the input point cloud.
4. Consider the plane correct if sufficiently many points are inliers.
5. If it is correct, re-estimate the plane for inliers and remove these inliers from the input point cloud.

The process is repeated in order to find the best planes.

The RANSAC algorithm has been extended in many ways and efficient variants are available: Schnabel et al. [74], for example, improved the original algorithm via hierarchical structuring and using normal attributes. Figure 6.1 shows examples of output of the RANSAC algorithm.

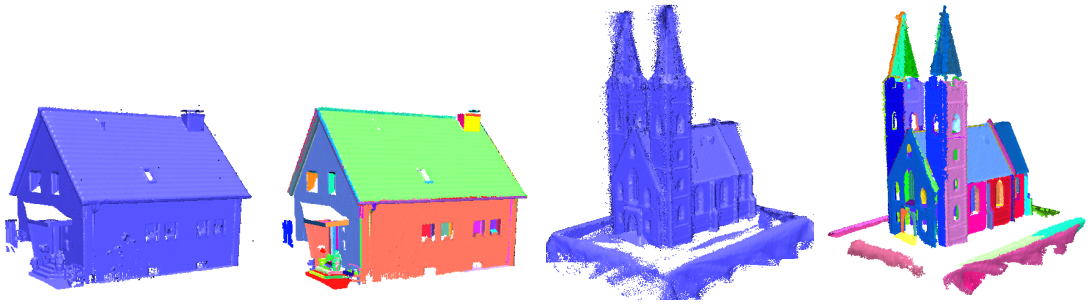


Figure 6.1: **RANSAC algorithm [74].** Original point clouds in blue, detected shapes in random colors.

Hough Transform

Some approaches based on planarity detection use accumulation spaces [15] or region growing for data sets containing a notion of connectivity, such as depth images [45]. Boulch and Marlet [17] use a Hough transform to estimate the planarity, with a discrete probability distribution of possible normals.

Their rationale is that the tangent plane – and therefore the normal vector – at a data point $\boldsymbol{p}$ can be well defined by a triplet of points in its neighborhood $N_K(\boldsymbol{p})$, but that picking all possible triplets would be unfeasible because of the huge complexity (on the order of $|N_K(\boldsymbol{p})|^3$). Therefore, only a subset is considered, and the sampling of triplets stops as soon as a confident decision can be made based on the empirical distribution in an accumulator.

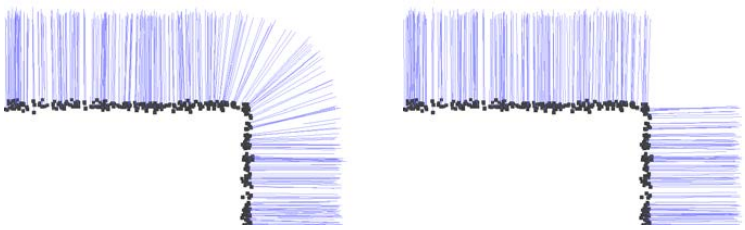


Figure 6.2: **Robust normal estimation [17].** Normal estimation with least squares regression (left) and with the robust normal estimation (right).

The output is depicted by Figure 6.2. Note that only large planar regions are typically

detected, as small scales cannot be reliably found through stochastic sampling.

Connecting the Primitives

Finding large planar regions is a "shape detection" problem. Reconstruction requires additional information such as adjacency relationships between primitives. Reconstruction algorithms typically constrain the planar regions to be simply connected [46], smooth [70], or globally regular [57, 84] to better capture the local geometry.

In order to perform reconstruction from a set of detected planar primitives, Jenke et al. [46] proceed by aligning and connecting the boundaries of adjacent primitives, while Schnabel et al. [73] extrapolate the primitives in order to reconstruct plausible surfaces on areas with missing data.

Reconstruction can also be formulated as an inside/outside cell labeling problem within a 3D adaptive space decomposition [22]. The use of space decomposition is usually robust and yields a valid embedding, but its complexity is high and the output meshes often unnecessarily complex.

Boulch et al. [16] introduce a regularization energy taking into account feature edges and corners during the process [16]:

$$E_{\text{regul}} = \lambda_{\text{area}} E_{\text{area}}(\boldsymbol{x}) + \lambda_{\text{edge}} E_{\text{edge}}(\boldsymbol{x}) + \lambda_{\text{corner}} E_{\text{corner}}(\boldsymbol{x}),$$

where the scalars λ are relative weights. The three terms of the energy provide the user with fine control over the output reconstruction:

- $E_{area}(\boldsymbol{x})$ penalizes the area of the reconstructed surface relatively to a scale s.
- $E_{edge}(\boldsymbol{x})$ penalizes the length of edges in the reconstructed surface.
- $E_{corner}(\boldsymbol{x})$ penalizes the number of corners in the reconstructed surface.

Note that the main objective is to provide trade simplicity of the output for fidelity to the input. This provides robustness to occlusions and missing data, putting a strong bound on the complexity of the output shape. An output of this algorithm is depicted by Figure 6.3.

Van Lankveld et al. [81] propose another constrained processing of primitives, extending the algorithm of Labatut et al. [54]. The rationale is to use a 3D constrained Delaunay tetrahedralization as a space decomposition, using the detected planes as constraints.

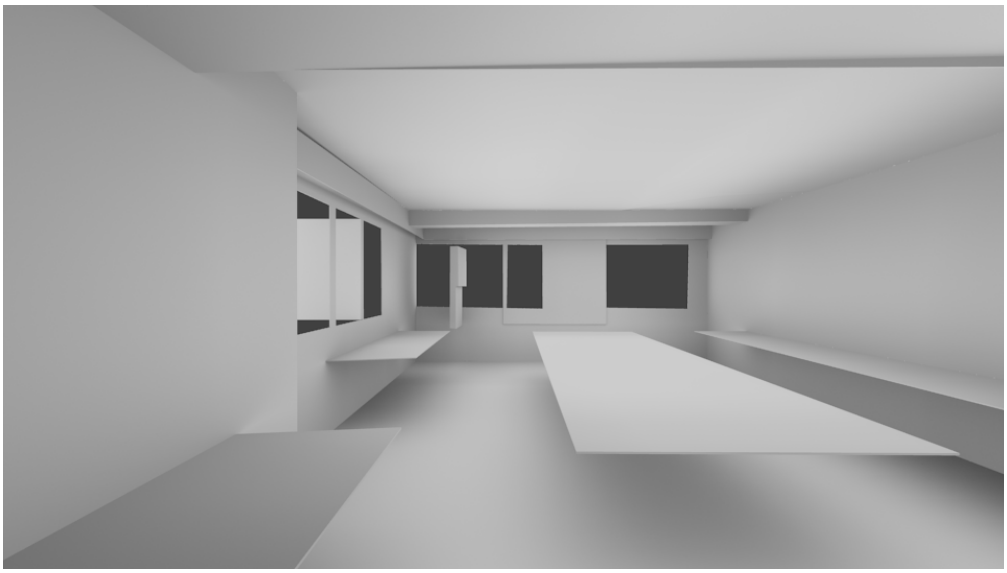


Figure 6.3: **Regularized algorithm [16].** The output of this algorithm is a coarse piecewise-planar mesh thanks to the regularization energy penalizing overly complex shapes.

Hybrid approaches have also been devised [55] to deal with both planar and free-form surfaces. The detected primitives are used to structure the point cloud via adjacency detection and are resampled within a Delaunay tetrahedralization.

Optimization

Finally, the connection between the primitives can be found through an optimization step. Schnabel et al. [73] propose an approach using the output of a shape detection algorithm such as RANSAC. They define a surface energy functional with the primitives shapes used as a guiding vector field, and minimize this energy with a graph-cut algorithm.

A similar rationale is used for the O-Snap [8] algorithm (Figure 6.4). Adjacency relationships among a set of primitives obtained through RANSAC are iteratively computed and enforced through a non-linear optimization. Using the set of adjacency relationships along with the positions of the boundaries of each primitive, an optimization step is performed to snap the polygons through an energy minimization. This method takes advantage of a combination of interactive modeling and automated algorithms: user interaction is crucial

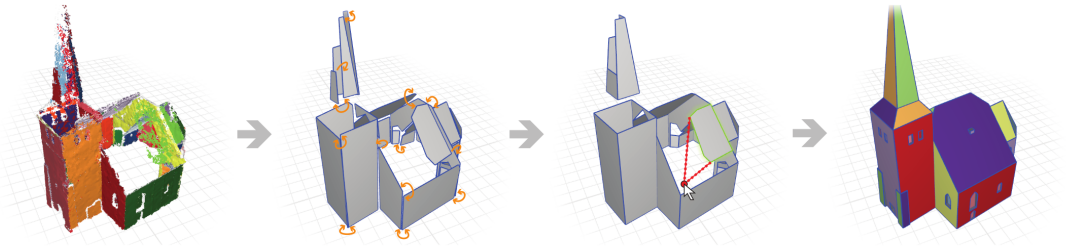


Figure 6.4: **O-Snap algorithm** [8]**.** Planes are detected via RANSAC; adjacencies are detected; polygons are snapped; the final reconstruction is user-assisted.

to generate satisfactory output from sparse input, as reconstruction from missing data is extremely ill-posed.

6.3 Feature-based reconstruction

Primitive-based methods often comprise a step of primitive connection to recover the features. Reconstruction of piecewise-planar surfaces can also be achieved the other way around, by detecting first the sharp features of the input (boundaries, sharp creases, corners) before deducing planar regions connecting these features.

Detection of sharp features is a separate topic of interest that can also be applied as a post-process of a smooth reconstruction algorithm. As a consequence, many approaches have been proposed over the years [43, 69, 27, 25, 63].

Feature Detection

The basic feature elements for piecewise-planar surfaces are represented by edges and vertices, although all vertices and edges are not necessarily related to sharp features. The first approach for reconstructing a shape from sharp features is to detect the main feature lines of a scene.

From a set of detected feature lines, Jenke et al. [47] reconstruct a feature-aware surface triangle mesh. More specifically, they represent the surface with a graph of local patches and mesh the union of all patches. The mesh is optimized via an energy function trading data fitting for smoothness while preserving the feature lines.

A variant of point set surfaces has been proposed by Oztierli et al. [67], using non-linear kernel regression. Local kernel regression (LKR) is a supervised regression method that approximates an unknown function $f(\boldsymbol{q_i}) : \mathbb{R}^d \mapsto \mathbb{R}$ at the neighborhood of an evaluation point $\boldsymbol{q}$ by the terms of a Taylor expansion:

$$f(\boldsymbol{q}) \approx f(\boldsymbol{q}) + (\boldsymbol{q_i} - \boldsymbol{q})^T \Delta f(\boldsymbol{q}) + \frac{1}{2}(\boldsymbol{q_i} - \boldsymbol{q})^T \mathbf{H} f(\boldsymbol{q})(\boldsymbol{q_i} - \boldsymbol{q}) + \dots,$$

where $\mathbf{H}f(\boldsymbol{q})$ denotes the Hessian matrix of $f(\boldsymbol{q})$. Using a normal constraint $\Delta f(\boldsymbol{p_i}) = \boldsymbol{n_i}$, the first order LKR minimization leads to the following function:

$$f(q) = \frac{\sum {\boldsymbol{n_i}}^T (\boldsymbol{q} - \boldsymbol{p_i})\theta(\boldsymbol{q})}{\sum \theta(\boldsymbol{q})},$$

where θ denotes a scale-dependent weight function. This is the definition of Implicit Moving Least Squares [52].

This formulation assumes the data to follow a smooth model and is thus not robust to sharp features. A robust LKR is devised using an outlier-robust estimator. The main rationale is to consider the sharp features as outliers in the sense of the normal vectors distribution and to detect the corresponding feature points. An iterative minimization of a Robust Implicit MLS is deduced:

$$f^k(q) = \frac{\sum n_i^T(q-p_i)\theta_i(q)w(r_i^{k-1})w_n(\Delta n_i^{k-1})}{\sum \theta_i(q)w(r_i^{k-1})w_n(\Delta n_i^{k-1})},$$

where w_n measures the difference between a predicted gradient and a sample normal, r_i^{k-} the residuals of the previous iteration and w a weight function.

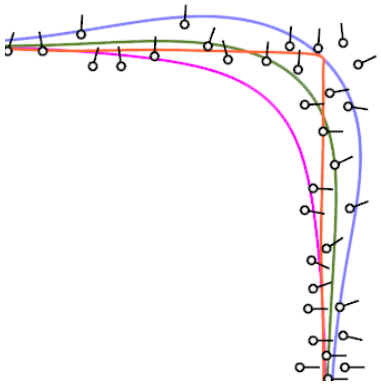


Figure 6.5: **Non-linear kernel regression [67].** The feature-aware version of MLS is given in red. IMLS is given as comparison in blue.

Figure 6.5 depicts a comparison of this type of MLS with the common IMLS.

Salman et al. [72] first detect feature points and then convert them into feature lines. Reconstruction is achieved through the combination of a weighted Delaunay triangulation to preserve features and an implicit surface reconstruction over smooth areas.

Finally, Dey et al. [28] propose a unified approach to handle all kinds of features including non-manifolds, via a combination of the Gaussian weighted graph Laplacian and the Reeb graph. A feature-preserving variant of the Cocone reconstruction method is then applied to produce the output shape.

Optimal Transportation

Low polygon-count reconstructions are also devised through feature-aware mesh simplification. A robust approach to reconstruct shapes with sharp features is to define a robust and two-sided error metric that preserves boundaries and sharp features.

De Goes et al. [26] take advantage of optimal transportation distances to robustly reconstruct curves with sharp angles. Edges of the Delaunay triangulation of the input point cloud are iteratively collapsed as long as the transport cost from the points to their support edges remains under a user-specified threshold. The position of sharp features is optimized using the same optimal transportation based metric (see Figure 6.6).

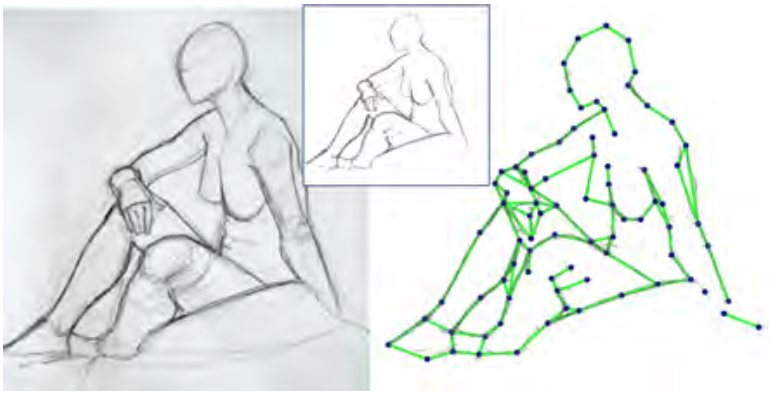


Figure 6.6: **Optimal transportation.** Left: input point set. Right: reconstruction with sharp features.

Digne et al. [31] extended this method to surfaces in 3D. The rationale is to simplify an initial 3D tetrahedralization of the input point cloud, guided by a similar optimal mass transport error metric. This approach is robust to both noise and outliers. Feature preservation is achieved by transporting the input points onto vertices, edges or faces. Although this approach is promising, solving for the optimal transport plan in 3D is too computationally expensive to obtain a practical algorithm.

6.4 Variational methods

Variational methods are devised using robust norms. The use of the L^1-sparse norm was proposed by Avron et al. [11]. It consists in finding a linear combination of basis functions through L^1 minimization. The problem is considered as a signal reconstruction process: a signal u is reconstructed using a linear combination of basis functions $phi_i \in \mathbb{R}^n$, $i = 1, ..., m$ with $m > n$ the number of points. This boils down to finding coefficients α_i such that:

$$u = \sum_{i=1}^{m} \alpha_i \phi_i.$$

From the Nyquist sampling theorem, it is known that the density of samples depends on the bandwidth. Inside the bandwidth, it can be dense and contain information in all frequencies. It has been proven since then [20, 33] that when only a small set of frequencies are present, as a result of a smoothness assumption, the number of samples required can decrease using L^1 minimization. This leads to a sparse representation:

$$\min_{\alpha} \|\alpha\|_0 \text{ s.t. } u(t_j) = \sum_{i=1}^{m} \alpha_i \phi_i(t_j),$$

where samples are given at times t_j and the zero norm $\|.\|_0$ measures the number of non-zero elements of a vector. Because solving this problem is highly non-convex, a common approximation is to replace the norm L^0 by the convex L^1 norm. In a similar way that a median operator is more robust to outliers than an averaging operator, the L^1 norm handles sharp features more robustly than the L^2 norm that produces smooth solutions.

From a geometric point of view, a point cloud with normals is reconstructed in a piecewise smooth fashion with a sparse set of singularities at sharp edges. A pairwise normal difference is computed to estimate shape smoothness: at crease angles, the distance between the normals of two distinct smooth subparts $\boldsymbol{n_i}$ and $\boldsymbol{n_j}$ is above a small threshold τ. Therefore, the normal field should have only a small number of local normal differences that are large.

The angle between the average normal of two points and their support line is computed: $\boldsymbol{n_{ij}} \cdot (\boldsymbol{p_i} - \boldsymbol{p_j})$. It is expected to be close to zero in general and non-zero near sharp features.

The normal field of the inferred shape is consistently estimated by minimizing a global

weighted L^1 penalty function $C^{\mathcal{N}}$:

$$C^N(\mathcal{N}, W, E) = \sum_{(\boldsymbol{p_i}, \boldsymbol{p_j}) \in E} w_{ij} c_{ij}^{\mathcal{N}}(\mathcal{N}),$$

where $W = \{w_{ij}\}$ denotes a set of weights defined to achieve sparsity, E an adjacency set and $\mathcal{N}$ the set of normal vectors.

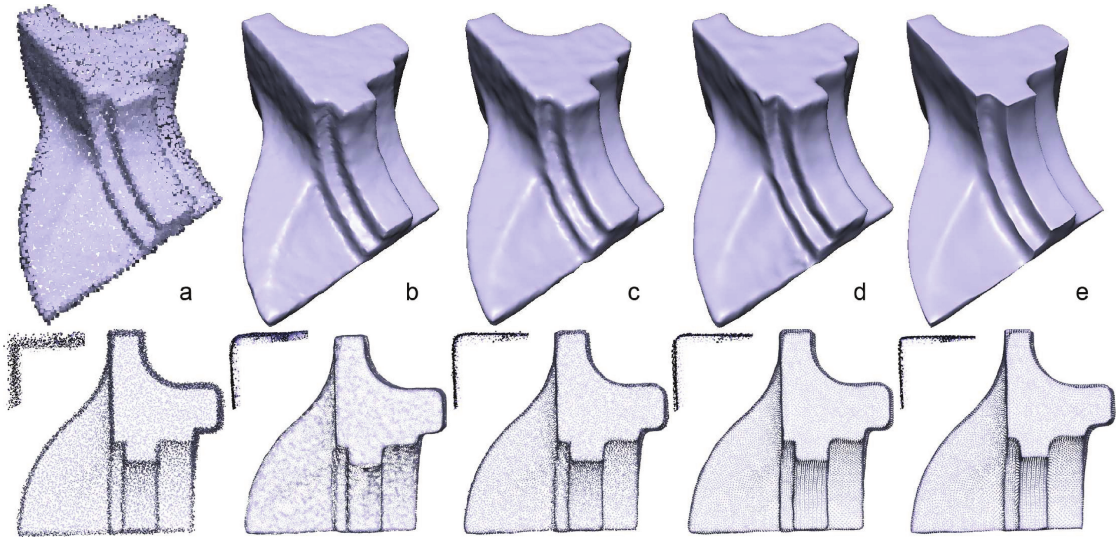


Figure 6.7: **Comparison of the L^1-sparse method [11].** (a) Input point cloud. (b) LOP [59] (c) DDMLS [58] (d) RIMLS [67] (e) L^1-sparse method.

This method robustly detects sharp features as discontinuities in the normal field by assuming the sparsity of these discontinuities. A comparison with other methods is depicted by Figure 6.7.

Note that the sparsity of the L^0 norm has also been applied to denoising [44, 24]. However, these robust norms often imply numeric-intensive methods and are thus not scalable. In addition, output shapes cannot represent non-manifold features, which makes these approaches not suited to capture more general shapes.

6.5 Conclusion

Reconstruction of piecewise-planar surfaces cannot be tackled using the same assumptions used for smooth reconstruction. The output of these two classes of method differ both in nature and purpose. The apparent simpler nature of the output – a set of connected planes – implies in fact a wider range of scientific challenges and is still an open problem.

Piecewise-planar reconstruction requires finding a balance between feature detection, conforming meshing and fidelity to the data. Some methods take advantage of the strong reliability of shape detection algorithms such as RANSAC as a first processing step: the problem boils down to connecting the detected planar primitives. This step can be achieved through minimizing energies that regularize the output mesh. In most cases, additional constraints are added to the output shape in order to reduce the space of solution (smoothness, low complexity first, etc.).

Some methods have also been devised to tackle the problem by detecting sharp features then building a mesh around them. This is achieved through metrics sensitive to discontinuities of the normal vector field or through graph-based approaches. Reconstruction is also achieved through simplifying a mesh with respect to these estimated constraints, using e.g., Delaunay triangulations.

Finally, some variational methods take advantage of the robustness of some specific norms to sharp features, but are often limited by the high computational complexity.

Most methods use strong assumptions on the inferred shape at the cost of a loss of generality: for instance, only few methods handle non-manifold surfaces. Although shape detection algorithms offer a solid ground to reconstruction of piecewise-planar surfaces, they are not always adapted to scenes containing both large planar shapes and many small details.

7

Problem Statement

We now consider input point clouds with no additional measurement and possibly ridden with noise and outliers. Several assumptions are made for the inferred surface.

Boundaries

The inferred shape is not required to separate an outside from an inside volume. It can furthermore contain an arbitrary number of boundaries.

Our algorithm is to some extend robust to variable sampling density under a user-specified threshold. However, data completion is not achieved and the boundaries near missing data result in boundaries on the output shape.

Piecewise-planar

The inferred shape is assumed to be planar almost everywhere and may contain sharp features. Input point clouds sampled on a piecewise-smooth surface are approximated by piecewise-planar reconstruction.

Figure 7.1 shows plausible reconstructions from a variety of input point clouds.

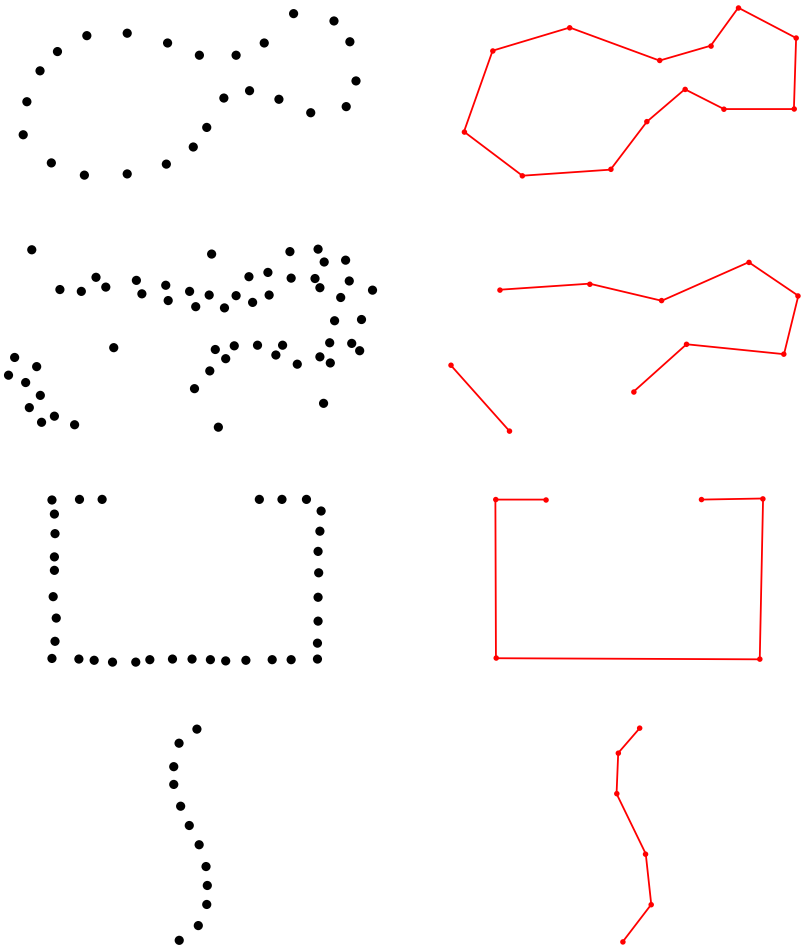


Figure 7.1: **Plausible reconstructions.** Left column: input point clouyd. Right column: plausible piecewise-planar reconstruction. Comprehensiveness is gained at the cost of losing resilience to missing data (compared with Figure 3.1).

8 Background

8.1 A priori Knowledge

8.1.1 Dimension

Our *dimension a priori* differs from Section 4.1 as we wish to reconstruct surfaces with non-manifold features and boundaries. The main assumption is that the inferred shape is formed by a connected set of planar regions.

Our geometry assumption is that every atomic part of the inferred shape is either a plane or has a sufficiently low curvature so to be decomposed as a set of planes.

8.1.2 Geometry

We do not rely on an implicit function to reconstruct a closed object but reconstruct instead multiple primitives with adjacency information. To make the problem better-posed, we thus need another transient representation. We make the assumption that the input point cloud can be effectively approximated by a sparse union of primitives matching subsets of the input.

Consider a subset $\mathcal{P}_i$ of the input point cloud $\mathcal{P}$. As discussed above, the surface is assumed to be decomposable into planar subparts. We thus define a *planar* property: a primitive lies on a *support plane* S_i formed by the two largest eigenvectors of the *principal component analysis* (PCA) of its associated point subset $\mathcal{P}_i$.

A plane only reflects the general orientation of the subset, but is by definition infinite. To further localize the subset in space, we add a property of *convexity*. Convexity offers stability and simplification. It has been thus largely discussed in literature, whether for efficient region selection [10] or for form feature recognition [51]. In our context, we compute the 2D convex hull $\mathrm{CH}_{S,i}$ of the orthogonal projection of the points of $\mathcal{P}_i$ onto the support plane S_i. This hull is stored as a cyclic list of *extreme points*, where each pair of consecutive extreme points is a *hull edge*.

The resulting planar convex primitive, denoted $\Pi_i \equiv \{S_i, \mathrm{CH}_{S,i}\}$, is made statistically reliable through PCA. Moreover, it offers a clear delineation of the convex region covered by the samples $\mathcal{P}_i$, which are now, by construction, inliers of Π_i.

This combination of *planarity* and *convexity* properties is crucial to both the efficacy and robustness of our algorithm. Using convex primitives allows for simple data structures and fast computations, and the potential adjacencies between primitives are by default simply connected.

8.2 Approximation Metric

We defined above a primitive associated to a point subset. We wish to measure the faithfulness of this primitive to the input point cloud and to define the approximation error between the reconstructed surface and the input.

8.2.1 Symmetric Hausdorff Distance

The Hausdorff distance is a convenient means to enforce a symmetric and well-posed geometric measure of reconstruction. Computing it exactly is hower difficult and time consuming, which is why it is often abandoned to the profit of relaxed metrics. We contribute an algorithmically efficient approach to its computation and adopt this Hausdorff metric, as it provides us with a reliable quality measure of the approximation.

We define the symmetric Hausdorff error metric between a primitive Π_i and its inliers $\mathcal{P}_i$ as:

$$d_{\mathrm{H}}(\Pi_i, \mathcal{P}_i) = \max\{ \sup_{a \in \Pi_i} \left[\inf_{b \in \mathcal{P}_i} d(a,b) \right], \sup_{b \in \mathcal{P}_i} \left[\inf_{a \in \Pi_i} d(a,b) \right] \},$$

where $d(a,b)$ denotes the Euclidean distance.

8.2.2 Properties

This metric is the maximum of two distinct quality criteria of the match between points in $\mathcal{P}_i$ and the primitive:

- $H(\mathcal{P}_i, \Pi_i) = \sup_{b \in \mathcal{P}_i} \inf_{a \in \Pi_i} d(a,b)$ is the maximum distance from $\mathcal{P}_i$ to its associated primitive: it measures a *level of noise*.
- $H(\Pi_i, \mathcal{P}_i) = \sup_{a \in \Pi_i} \inf_{b \in \mathcal{P}_i} d(a,b)$ is the maximum distance from the primitive to its inliers: it this sense, it is related to the *sampling density*.

Selecting upper bounds on these two criteria thus provides control over the maximum level of noise a primitive tolerates, and over the minimum density of points required for the point subset to be geometrically relevant.

We assume that knowledge on the acquired point cloud is sufficient to estimate a bound ε_N on noise, and a bound ε_S on sampling density.

9 Contribution

9.1 Approximations

Our algorithm relies on both the primitives and their approximation error. We define approximations that guarantee a reasonable computation complexity while keeping the algorithm reliable.

9.1.1 Convex Hull

Although the convex hull is conceptually simple, its computational complexity is dependent on the associated point subset it matches: a set of aligned points on a segment are for example enclosed in a convex hull with 2 extreme points: the end points of the segment. If this point set is noisy, a potentially high number of additional extreme points may be required.

In order to provide the user with better control over the complexity of the approximation and to speed up the computations, we approximate the convex hull by *discretizing the space of directions*. Consider a point subset $\mathcal{P}_i$ and a random direction $\boldsymbol{d}$ on space. We denote by $\boldsymbol{p_{min}}$ and $\boldsymbol{p_{max}}$ the two extreme points of $\mathcal{P}_i$ along this direction.

Using the definition of a convex hull, we make the following observations:

1. The segment joining $\boldsymbol{p_{min}}$ and $\boldsymbol{p_{max}}$ is inside the convex hull of $\mathcal{P}_i$.

2. The convex hull of $\mathcal{P}_i$ is located between the planes P_{min} and P_{max}, orthogonal to $\boldsymbol{d}$ and passing respectively through $\boldsymbol{p_{min}}$ and $\boldsymbol{p_{max}}$.

The segment $\{\boldsymbol{p_{min}}; \boldsymbol{p_{max}}\}$ is a lower approximation of the convex hull while the half-space between P_{min} and P_{max} is an upper approximation. Using d directions $\boldsymbol{d}_j$, we get d pairs of points $\boldsymbol{p^j_{min}}$ and $\boldsymbol{p^j_{max}}$ and d pairs of planes P^j_{min} and P^j_{max}. We make the following observations:

1. The convex hull of the union of all points $\boldsymbol{p^j_{min}}$ and $\boldsymbol{p^j_{max}}$ is included in the convex hull of $\mathcal{P}_i$.

2. The convex hull of $\mathcal{P}_i$ is located in the intersection of all half-spaces between all pairs of planes $\{P^j_{min}; P^j_{max}\}$.

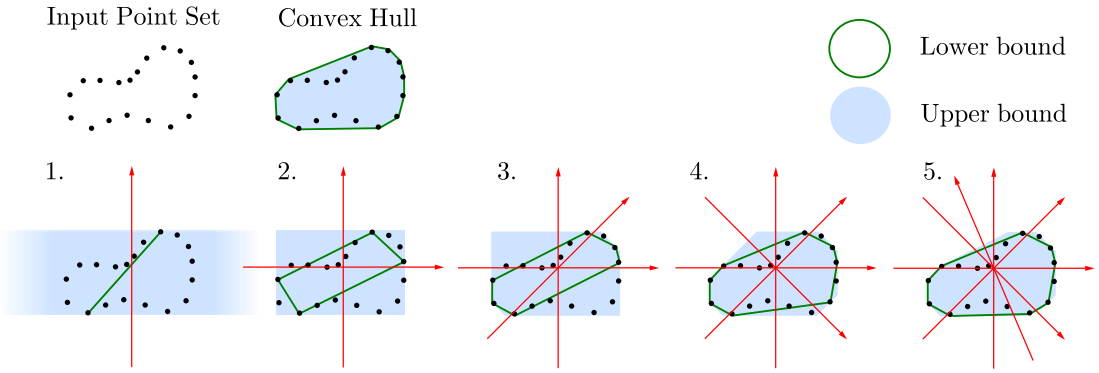


Figure 9.1: **Convex hull approximation.** Evolution of the lower and upper bounds of the approximated convex hull for 1 to 5 directions.

Figure 9.1 depicts how these lower and upper approximations evolve when increasing the number of directions.

This method allows for fast computation of convex hulls. Moreover, it limits the number of extreme points of the approximated hulls: using d direction, the lower approximation of the convex hull of a random point subset has at most $2d$ distinct extreme points. In our experiments, we found that 50 directions are sufficient to approximate convex hulls while keeping the most significant details.

9.1.2 Error Computation

We contribute an algorithm to efficiently compute the Hausdorff distance. Using convex primitives allows for efficient computation through a decomposition of the primitives into triangles.

We enforce a limit on $H(\mathcal{P}_i, \Pi_i)$ on each primitive by construction: the inliers $\mathcal{P}_i$ of a primitive Π_i are restricted to the points from $\mathcal{P}$ that are within a ε_N-thickened version of the $2D$ convex hull $\mathrm{CH}_{S,i}$ of this primitive.

Since points from $\mathcal{P} \backslash \mathcal{P}_i$ are necessarily further than ε_N-away from Π_i, we replace the evaluation of $H(\Pi_i, \mathcal{P}_i)$ by:

$$H(\Pi_i, \mathcal{P}) = \sup_{\boldsymbol{a} \in \Pi_i} \inf_{\boldsymbol{b} \in \mathcal{P}} d(\boldsymbol{a}, \boldsymbol{b}).$$

We thus compute the maximum distance from the primitive to the whole input point cloud $\mathcal{P}$. The problem boils down to finding the point of the convex hull $\mathrm{CH}_{S,i}$ that is the farthest away from $\mathcal{P}$.

Bounds

Each primitive Π_i is represented as a 2D Delaunay triangulation embedded in 3D space on the support plane S_i. We reformulate the Hausdorff distance as:

$$H(\Pi_i, \mathcal{P}) = \max_{\mathcal{T}} H(\mathcal{T}, \mathcal{P}),$$

where $\mathcal{T}$ denotes a cell of the triangulation.

The problem hence reduces to computing the Hausdorff distance from each triangle $\mathcal{T}$ partitioning the convex hull CH_i to the point cloud $\mathcal{P}$. For each such triangle $\mathcal{T}$, we store precomputed values $d_k = d(\boldsymbol{v_k}, \mathcal{P})$, $k = 0, 1, 2$ on its vertices, using a precomputed k-d tree data structure over $\mathcal{P}$ for efficient closest point queries.

By definition of the Hausdorff distance, a lower bound of $H(\mathcal{T}, \mathcal{P})$ is defined as follows:

$$\underline{H}(\mathcal{T}, \mathcal{P}) = \max_{k \in 0,1,2} \{d_k\}.$$

Since the Euclidean distance is 1-Lipschitz, an upper bound is also computed:

$$\overline{H}(\mathcal{T},\mathcal{P}) = \max_{p\in\mathcal{T}} \min_{k\in 0,1,2} \{d_k + d(\boldsymbol{v_k}, \boldsymbol{p})\}.$$

This upper bound is reached at a specific point $\boldsymbol{p}$: the solution to the Apollonius problem [38] that coincides with the center of the circle internally tangent to the 3 circles centered at vertices $\boldsymbol{v_k}$ and with respective radius $r_k = d_k - \min_{k\in 0,1,2}\{d_k\}$.

When this Apollonius center lies outside $\mathcal{T}$, we instead bisect $\mathcal{T}$ by inserting a vertex on the edge of $\mathcal{T}$ visible from the Apollonius center. These two cases are depicted by Figure 9.2.

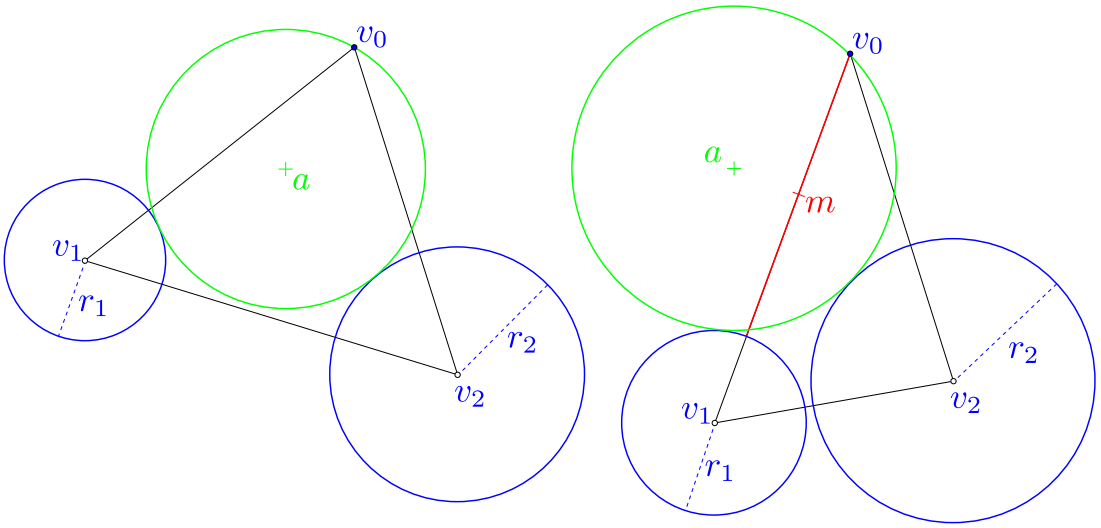


Figure 9.2: **Apollonius center.** Optimal upper bound locations depicted with $d_0 = \min_{k\in 0,1,2}\{d_k\}$. $r_1 = d_1 - d_0$ and $r_2 = d_2 - d_0$. Left: Apollonius center $\boldsymbol{a}$ inside the triangle. Right: when the Apollonius center $\boldsymbol{a}$ lies outside the triangle, the middle $\boldsymbol{m}$ of the red segment is taken instead.

From the two bounds $\underline{H}$ and $\overline{H}$, we define the approximated value of $H(\mathcal{T},\mathcal{P})$:

$$\tilde{H}(\mathcal{T},\mathcal{P}) = (\underline{H}(\mathcal{T},\mathcal{P}) + \overline{H}(\mathcal{T},\mathcal{P}))/2,$$

and the maximum error of approximation:

$$e_H(\mathcal{T},\mathcal{P}) = \overline{H}(\mathcal{T},\mathcal{P}) - \underline{H}(\mathcal{T},\mathcal{P}).$$

Recursive Subdivision

Recursively subdividing $\mathcal{T}$ reduces the approximation error, either via trisection by inserting the Apollonius center inside the triangle and connecting it to the triangle vertices, or via bisection by inserting the corresponding point on an edge and connecting it to the facing vertex (see Figure 9.2). The error is now defined as $H(\mathcal{T}, \mathcal{P}) = \max_j\{H(\mathcal{T}_j, \mathcal{P})\}$, where T_j denotes a subdivided triangle, and therefore: $H(\Pi_i, \mathcal{P}) = \max_j\{H(\mathcal{T}_j, \mathcal{P})\}$.

Consequently, by recursively subdividing the triangulation of CH_i until all triangles have $e_H(\mathcal{T}_j, \mathcal{P})$ under a specified maximum tolerance error, we approximate $H(\Pi_i, \mathcal{P})$ within the tolerance. Each triangle $\mathcal{T}_j$ is stored in a priority queue sorted by $\overline{H}(\mathcal{T}_j, \mathcal{P})$. We iteratively pop triangles from the queue and subdivide them while $e_H(\Pi_i, \mathcal{P}) < e_{limit}$ as described above; d_k is computed only for the new vertices inserted by subdivision.

Given a user-specified error tolerance ε_S on the Hausdorff distance $H(\mathcal{T}_j, \mathcal{P})$, this algorithm takes advantage of early ending cases.

9.2 Fine-to-Coarse Algorithm

We first detail a preliminary approach where planar primitives are grown hierarchically using a dynamic priority queue. The algorithm is initialized with a trivial solution: each point of the input point cloud generates a pointwise primitive with a zero Hausdorff distance. The rationale is to simulate primitive merging operation and iteratively applying the operation that increases the least the Hausdorff distance, as long as the distance is lower than a user-specified error tolerance.

9.2.1 Atomic Operations

We devise a fine-to-coarse algorithm by successive primitive merging, starting with pointwise primitives. A primitive may end up covering large areas of the scene, resulting in a very large amount of operations. We discuss next an efficient way to compute these operations.

Merging Support Planes

The support plane S_i of a primitive Π_i is computed through principal component analysis (PCA). PCA is based on an eigendecomposition of the covariance matrix of the considered point subset $\mathcal{P}_i$:

$$M = \begin{pmatrix} \sum(p_x - c_x)^2 & \sum((p_x - c_x) \cdot (p_y - c_y)) & \sum((p_x - c_x) \cdot (p_z - c_z)) \\ \sum((p_x - c_x) \cdot (p_y - c_y)) & \sum(p_y - c_y)^2 & \sum((p_y - c_y) \cdot (p_z - c_z)) \\ \sum((p_x - c_x) \cdot (p_z - c_z)) & \sum((p_y - c_y) \cdot (p_z - c_z)) & \sum(p_z - c_z)^2 \end{pmatrix},$$

where c denotes the centroid of $\mathcal{P}_i$.

Computing the PCA of a point subset $\mathcal{P}_i$ of size n_i has a complexity of $O(n_i)$. The complexity for computing the PCA of two merged point subsets $\mathcal{P}_i$ and $\mathcal{P}_j$ is thus $O(n_i+n_j)$.

As we already computed the respective PCAs of $\mathcal{P}_i$ and $\mathcal{P}_j$, we can avoid recomputing the covariance matrix of the merged point subset $\mathcal{P}_k$. Using polynomial expansions, we decompose the covariance matrix M as follows:

$$M = \begin{pmatrix} \sum p_x^2 & \sum p_x \cdot p_y & \sum p_x \cdot p_z \\ \sum p_y \cdot p_x & \sum p_y^2 & \sum p_y \cdot p_z \\ \sum p_z \cdot p_x & \sum p_z \cdot p_y & \sum p_z^2 \end{pmatrix} + \begin{pmatrix} \sum p_x \\ \sum p_y \\ \sum p_z \end{pmatrix} \cdot \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix}^T - \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix} \cdot \begin{pmatrix} \sum p_x \\ \sum p_y \\ \sum p_z \end{pmatrix}^T + N_i \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix} \cdot \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix}^T$$

We define matrices A, P and C as follows:

$$M = A + P \cdot C^T - C \cdot P^T + N_i \cdot C \cdot C^T.$$

The respective matrices A_i, P_i and C_i are stored with their respective primitive Π_i. We want to find the support plane of Π_k obtained by merging Π_i with Π_j. Computing the new centroid C_k from C_i and C_j is straightforward, using a sum weighted by the respective number of points n_i and n_j of the primitives. The other two matrices are obtained by $A_k = A_i + A_j$ and $P_k = P_i + P_j$.

Therefore, by storing the matrices A_i, P_i and C_i of each primitive Π_i, the complexity for computing the PCA of a primitive Π_k obtained by merging Π_i with Π_j drops to $O(1)$.

Merging Convex Hulls

As presented in Section 9.1.1, convex hulls are approximated by intervals on a set of discrete directions. Note that this gives an approximation of the *3D convex hull* $\mathrm{CH}^{3\mathrm{D}}$ that is projected onto the support plane S_i of primitive Π_i to compute the 2D convex hull CH_i.

For simplicity, we use the lower approximation of the convex hull, i.e., the set of at most 2 extreme points per direction. Using n directions, the approximated 3D convex hull is represented as follows:

$$\tilde{\mathrm{CH}}^{3\mathrm{D}}{}_i = \{p^0_{i,min}, p^0_{i,max}, p^1_{i,min}, p^1_{i,max}, ..., p^{n-1}_{i,min}, p^{n-1}_{i,max}\}.$$

Consider two approximated convex hulls $\tilde{\mathrm{CH}}^{3\mathrm{D}}{}_i$ and $\tilde{\mathrm{CH}}^{3\mathrm{D}}{}_j$. The convex hull $\tilde{\mathrm{CH}}^{3\mathrm{D}}{}_k$ obtained by merging $\tilde{\mathrm{CH}}^{3\mathrm{D}}{}_i$ with $\tilde{\mathrm{CH}}^{3\mathrm{D}}{}_j$ is found through a comparison of extreme points direction by direction:

$$\tilde{\mathrm{CH}}^{3\mathrm{D}}{}_k = \left\{ \min\left(p^m_{i,min}, p^m_{j,min}\right), \max\left(p^m_{i,max}, p^m_{j,max}\right) \right\}_{m \in [0, n-1]}.$$

This drastically reduces the computational complexity of the merging operations. More

specifically, the complexity is $O(d)$ where d denotes the number of directions. The output of the merging operation is the set of extreme points of the 3D convex hull of the union of the points covered by Π_i and Π_j. Projecting these points onto the merged support plane and computing the 2D convex hull of the resulting point set completes the primitive's construction.

9.2.2 Priority Queue

Now that we can efficiently merge primitives and compute the Hausdorff error (see Section 9.1.2), we define a fine-to-coarse algorithm that relies on a dynamic priority queue.

We initialize the algorithm with a trivial solution using one primitive per point. The initial error of approximation error is 0. An adjacency graph is built using a 3D Delaunay tetrahedralization of the input point cloud. For each edge between two primitives, we simulate the merging operations of the two primitives and use the resulting Hausdorff error as a priority criterion. More specifically, we give a higher priority to operations inducing low error.

Operations are stored in a modifiable priority queue. We pop the highest priority operation and apply it. When merging two primitives Π_i and Π_j, all the other operations involving one of these two primitives are invalid. We therefore keep track of all operations involving which primitive and discard them when two primitive are merged.

Having discarded all invalid operations, we also compute all newly possible merging operations. We update the adjacency graph and simulate merging the newly created primitive with its neighbor primitives. The algorithm stops when no more primitives can be merged without violating the user-specified tolerances ε_S or ε_N.

Figure 9.3 illustrates this algorithm at work on a synthetic point cloud.

9.2.3 Discussion

Although the set of primitives offers a sparse representation of the input scene with a significant gain in terms of data compression, several artifacts appear, as depicted by Figure 9.4. We now review these artifacts.

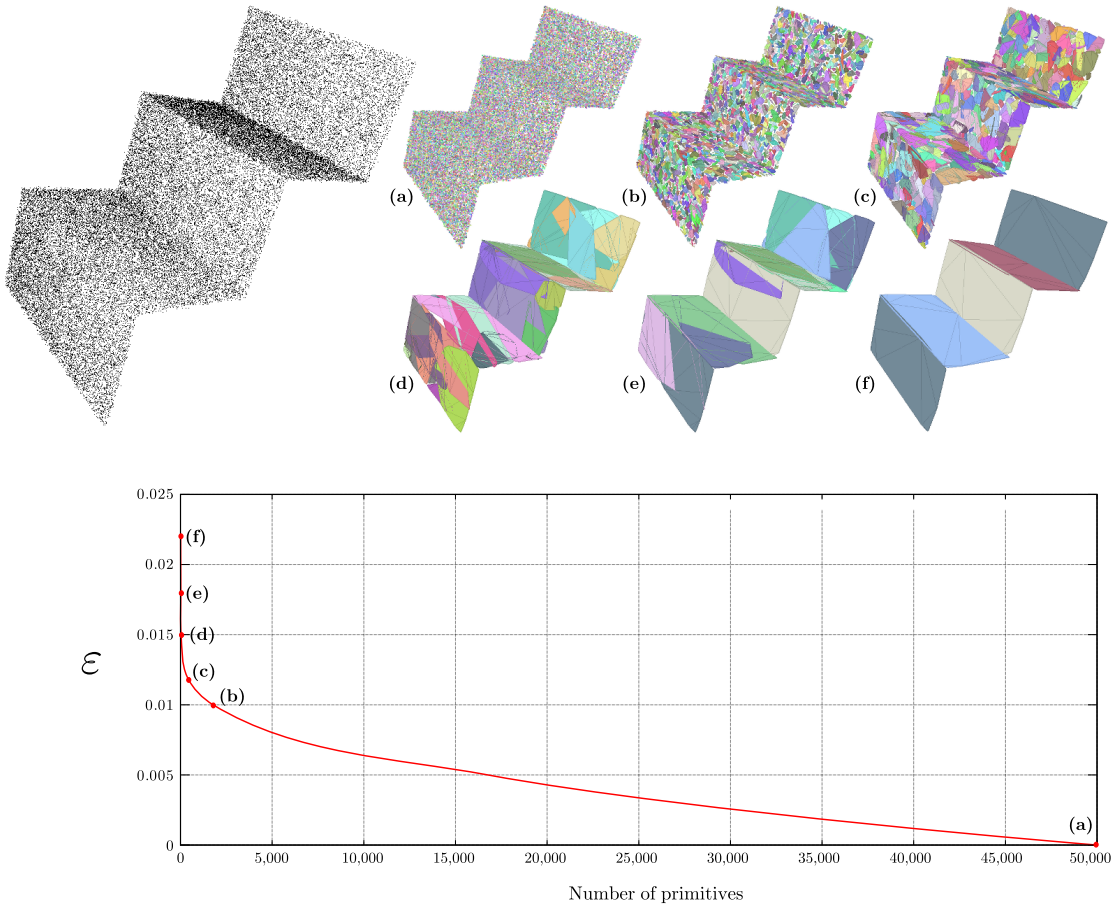


Figure 9.3: **Fine-to-coarse algorithm.** Bounds ε_S and ε_N are given the same value. Top: the algorithm iteratively merges primitives while increasing the error until 5 primitives remain. Bottom: we plot the symmetric Hausdorff error against the number of primitives. The input point cloud contains 50K points.

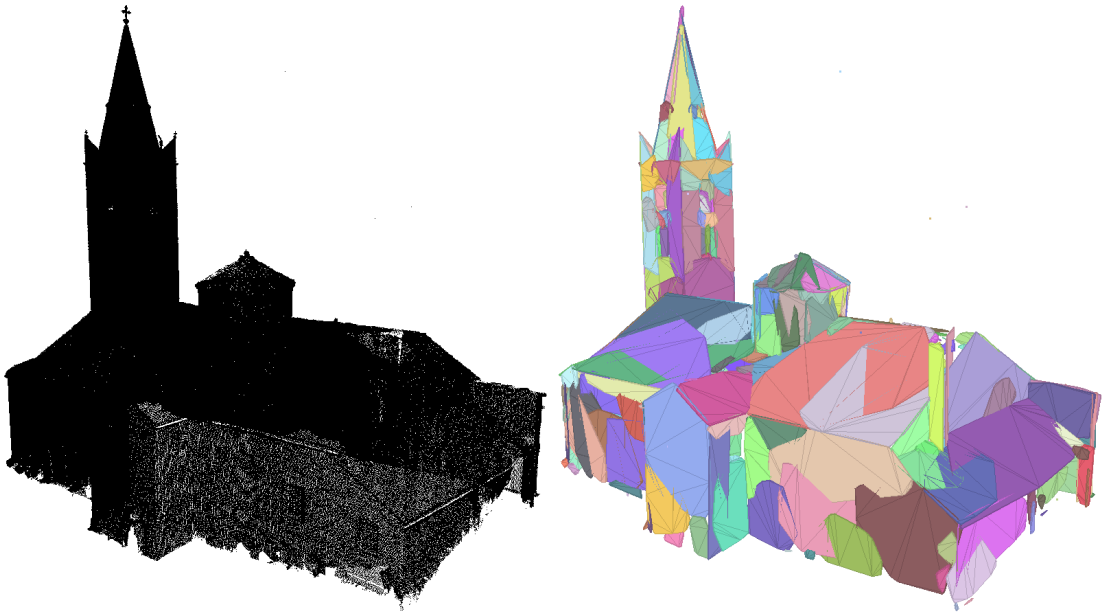


Figure 9.4: **Fine-to-coarse algorithm.** Point cloud & set of 364 primitives ($\varepsilon_S = \varepsilon_N = 0.010$).

Badly Placed Primitives

When a primitive is small, i.e., under the noise level, there is no way to distinguish if it covers a planar region or a sharp feature. As the algorithm is purely hierarchical, early errors are not fixed later on. Badly placed primitives may never grow larger than the feature they cover, producing a suboptimal set of primitives.

A first observation is that primitive should be initialized on planar parts and not on sharp features. In order to get a well-behaved set of adjacencies between primitives, sharp features should instead be the location of one or several primitives boundaries. This is consistent from a dimensional point of view: primitives are 2D objects and should therefore favor coverage of 2D parts of the surface, not of 1D features such as edges or of 0D features such as vertices.

Overlaps

The Hausdorff metric ensures that the data points are less than ε_N-away from the set of primitives and that no part of a primitive is more than ε_S-away from a data point. There is however no restriction on the number of times a data point can be covered. As a result, primitives may overlap on some parts of the shape, leading to an overlapping decomposition instead of partitioning.

Overlaps are an issue, as the primitive creation step should make the reconstruction a better-posed problem, and thus make it easier to generate an output mesh from the input point cloud. In addition, and although the primitives approximate well their associated point subset, the set of primitives is hardly usable for adjacency recovery.

Complexity

The algorithm is fully hierarchical and its complexity depends, among other things, on the number of points of the input point cloud. The same shape sampled with a higher density thus requires longer computation times, even if the total amount of details is low.

Note also that a large number of operations are simulated every time a primitive is created compared to the number of operations applied. Experimentally, less than 10% of the computed operations are applied.

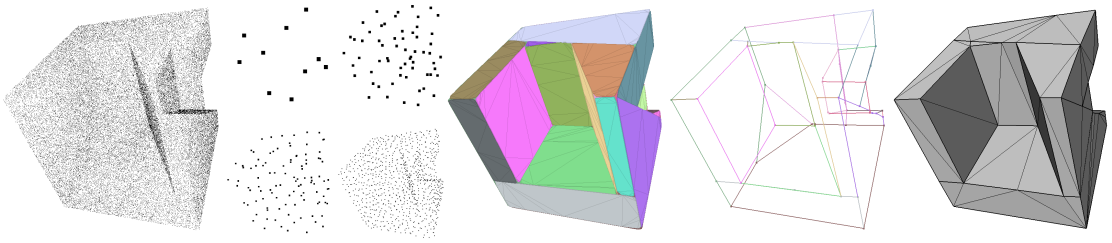


Figure 9.5: **Overview.** From left to right: input point cloud, multiscale point cloud, primitive decomposition, adjacency and output mesh.

9.3 Coarse-to-Fine Approach

Although the fine-to-coarse approach above described does not produce reliable results, we observed two important rationale:

- **Overlaps:** adjacency is hard to compute when subsets of the input point cloud are covered with overlaps. In order to facilitate adjacency recovery, primitives should overlap not more than the error tolerance. Ideally, the primitives should *partition* the input point cloud.
- **Sharp features:** primitives initialized close from sharp features exhibit lower robustness. Conversely, primitives initialized on large planar parts are more reliable and improve efficiency as they cover large parts of the input point cloud.

We next devise a coarse-to-fine approach.

Hierarchical Clustering

Similarly to Section 5.1.3, we generate multiple scales of the input point cloud $\mathcal{P}$ through hierarchical clustering [68]. Each cluster representative is chosen as the *closest point* to the cluster centroid among $\mathcal{P}$.

Scales are given indices i where index 0 refers to the input point cloud (finest scale). Denoting by s the size of each cluster (set by default to 5 in all shown experiments), the size of the i^{th} scale is $\frac{n}{s^i}$.

We denote by C be the covariance matrix of a cluster c and λ_0, λ_1 and λ_2 its three eigenvalues sorted in decreasing order. Pauly et al. [68] define a measure of surface variation for a cluster c:

$$sv(c) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}.$$

This measure ranges from 0 when all points of c are coplanar to 1/3 if points of c are isotropically distributed. We can thus detect the most planar clusters across scales.

Note that all Hausdorff error measures are performed with respect to the finest scale, as we only want to approximate the original point cloud. A single k-d tree is thus kept in memory for this particular scale. The multiscale representation is used as a support for primitive building.

The last preprocessing step if the computation of a 3D Delaunay tetrahedralization for the coarsest non-empty scale, used for adjacency purposes as in Section 9.2.

9.4 Constructing Primitives

We want to construct primitives both efficiently and reliably, starting on large planar parts and ending on smaller features. Each primitive is initialized from the coarsest scale and grown throughout all scales until the input point cloud (scale 0) is reached. As we want to partition the point cloud with our primitive set, each point or cluster representative covered by the primitive is removed from its scale. The k-d tree is updated every time points are removed from the input point cloud.

We proceed as follows until all scales are empty. For each primitive Π_i the region-growing algorithm proceeds as follows:

1. Initialize Π_i from the coarsest non-empty scale;
2. Go to the immediately finer scale (if any);
3. Find inliers of the supporting plane of Π_i at current scale;
4. Grow Π_i based on these inliers;
5. Remove all inliers of primitive Π_i found at current scale;
6. Go back to step 2 if points remain.

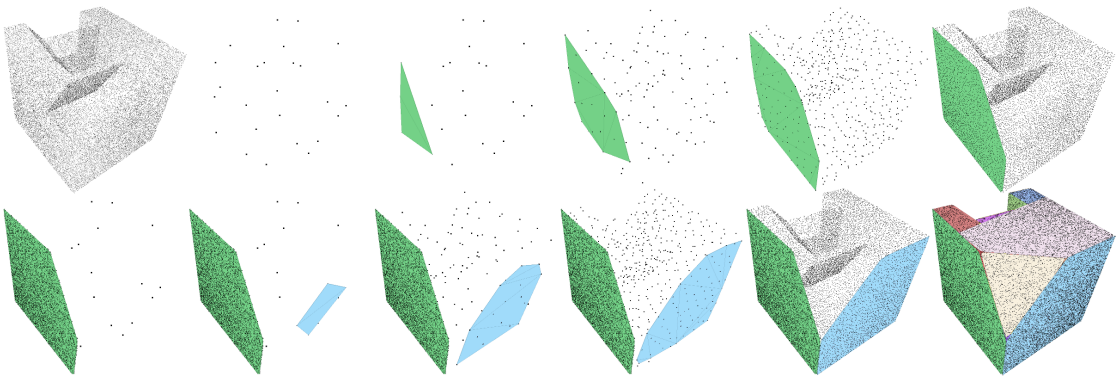


Figure 9.6: **Primitive construction.** Input point cloud; first primitive constructed throughout 4 scales; second primitive constructed throughout 4 scales; final set of primitives.

9.4.1 Initialization

Consider the coarsest non-empty scale of the multiscale representation of the input point cloud. A primitive Π_i is initialized using the point $\boldsymbol{p}$ of this scale, representative of the cluster of smallest *surface variation* $sv(p)$, i.e., the most planar cluster.

When a primitive is reduced to a single point, we define $d_{\mathrm{H}}(\Pi_i, \mathcal{P}_i) = 0$. The rationale is to add points in Π_i as long as its sampling error remains under the user-defined threshold ε_S. To this end, we simulate the addition of each unused neighbor point of Π_i in the precomputed 3D Delaunay tetrahedralization of the coarsest scale. We estimate the new best fitting plane of $\mathcal{P}_i$, compute the 2D convex hull and estimate $d_{\mathrm{H}}(\Pi_i, \mathcal{P}_i)$. At each step we grow Π_i if and only if $d_{\mathrm{H}}(\Pi_i, \mathcal{P}_i) < \varepsilon_S$, by adding the neighbor inducing the smallest error $d_{\mathrm{H}}(\Pi_i, \mathcal{P}_i)$.

When no more point can be added in Π_i without $d_{\mathrm{H}}(\Pi_i, \mathcal{P}_i)$ becoming higher than ε_S, the initialization is over. All points that are current inliers of Π_i are removed from the coarsest scale and the primitive goes to the immediately finer scale as we detail next.

Note that the removal of inliers may cause the 3D Delaunay tetrahedralization of this scale to become empty: this case triggers the construction of a 3D Delaunay tetrahedralization of the next coarsest non-empty scale.

9.4.2 Growing Within a Scale

We now consider the immediately finer scale. We want to grow Π_i further by including finer details of the inferred shape. For efficiency purpose, we consider in this step of computation that the support plane of Π_i is fixed: adding points to it only triggers an update of the 2D convex hull.

Inliers and Candidates

We identify the inliers of Π_i of this scale, i.e., all the points that are less than ε_N-away from the support plane S_i of Π_i and whose projection onto S_i lies within the 2D convex hull $\mathrm{CH}_{S,i}$. These inliers are covered by Π_i and therefore removed from the current scale.

We also seek points in the current scale that are candidates to the expansion of Π_i. As we consider S_i fixed, we only consider points that are less than ε_N-away from the

support plane S_i. We also use a simpler adjacency structure by computing a 2D Delaunay triangulation restricted to the projections of these candidate points.

Growing

Growing the primitives is performed in a similar way than during the initialization stage, with the significant difference that the plane is fixed and that the adjacency structure is a 2D Delaunay triangulation restricted to the candidate points instead of a 3D Delaunay tetrahedralization of the whole scale.

This allows for a faster computation of the new error $d_{\mathrm{H}}(\Pi_i, \mathcal{P}_i)$. Adding the projection onto S_i of a candidate point to the 2D convex hull $\mathrm{CH}_{S,i}$ only creates a few triangles in this hull. We can therefore limit the Hausdorff computation described in Section 9.1.2 to these new triangles. A candidate is included if the new error $d_{\mathrm{H}}(\Pi_i, \mathcal{P}_i)$ is lower than ε_S. Upon termination (i.e., when no candidates remain), we update the fitting plane S_i.

All points covered by Π_i are removed from the current scale. This growing process is then repeated throughout all scales. When scale 0 is reached, removing inliers triggers an update of the k-d tree used to compute Hausdorff distances.

9.4.3 Output

This initialization followed by the region-growing phase results in the construction of a convex primitive with its support plane, convex hull, and inliers. Repeating this process until all scales are empty produces an output primitive set that approximates the input point cloud under the given tolerances.

The input point cloud is segmented into non-overlapping primitives by construction. Figure 9.6 illustrates the growing process.

9.5 Detecting Adjacency

The output from previous step is a set of disconnected primitive that are both convex and planar. Some adjacency relationship must be inferred before generating a conforming mesh.

9.5.1 Sampling Hull Boundaries

Remind that a primitive Π_i is a 2D convex hull embedded in a supporting plane. We define the boundary of this primitive as the set of segments joining two consecutive extreme points of $\mathrm{CH}_{S,i}$.

We densely sample this boundary with an ordered cyclic list of *witness* points. These witnesses are spaced out by a distance of at most ε_S. The n^{th} witness of primitive Π_i is denoted by $\boldsymbol{w}_n^i$, see Figure 9.7, top left.

9.5.2 Detecting Adjacency

A witness $\boldsymbol{w}_n^i$ of primitive Π_i is labeled inlier of another primitive Π_j if it is closer than ε_S from primitive Π_j. Each primitive being bounded and planar, a primitive Π_i can only be adjacent to another primitive Π_j if at least one of its witness points is an inlier of the other primitive, or if a witness point of the other primitive is one of its inliers.

We compute adjacency relationships between all primitives by iterating over each primitive Π_i and over each of its witnesses $\boldsymbol{w}_n^i$. A witness is tagged as part of a primitive when listed as one of its inliers.

Figure 9.7 (middle top) shows the adjacency detection step.

9.5.3 Inserting Corners

As a consequence of convexity, the intersection between 2 primitives is simply connected. Estimating the shape of the adjacencies between primitives from the set of witness-adjacencies is therefore facilitated.

We observe two cases, depicted by Figure 9.8:

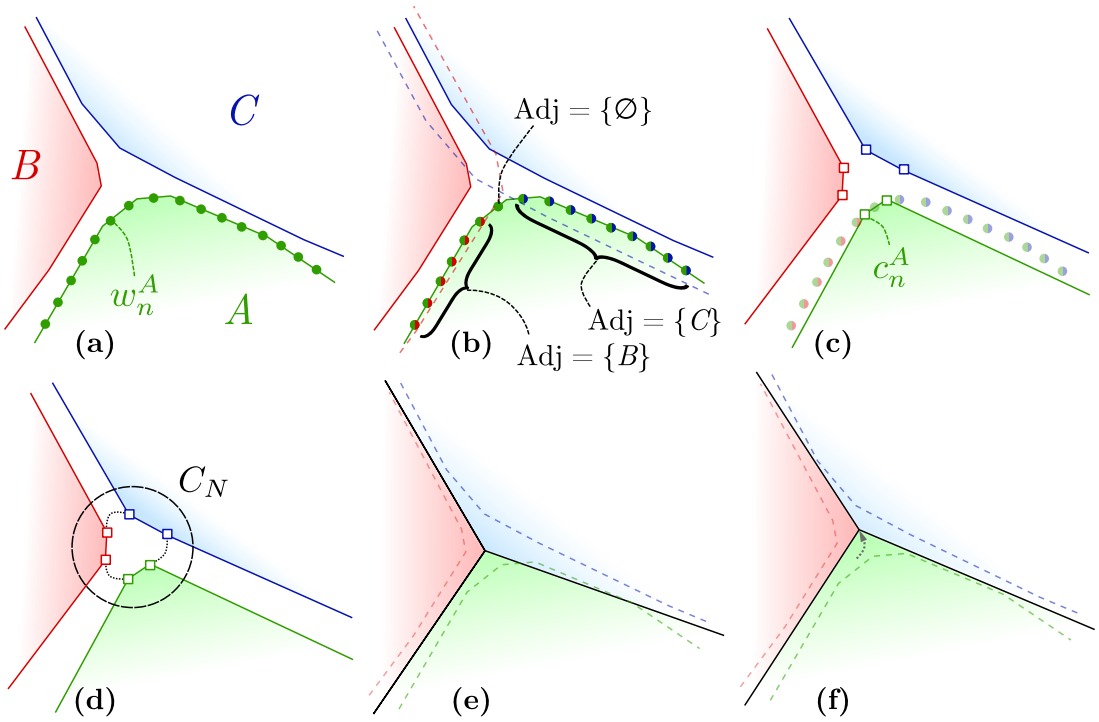


Figure 9.7: **Detecting adjacencies.** We recover the adjacency relationships between primitives A, B and C (focusing on A). **(a)** Sampling hull boundaries. **(b)** Detecting adjacency. **(c)** Corner placement. **(d)** Corner clustering. **(e)** Snapping. **(f)** Optimization.

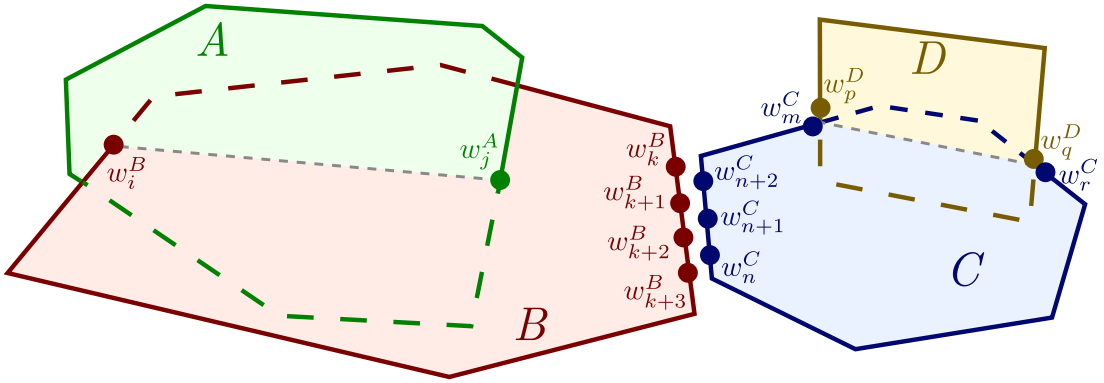


Figure 9.8: **Intersections between primitives.** Primitives A and B will be connected by a non-manifold edge joining two witness points belonging to each of them. Primitives B and C will be connected by a manifold edge. Primitives C and D will be connected by a non-manifold edge joining 2 clusters of witness points of both primitives.

- Boundary adjacency: primitives Π_i and Π_j share a simply-connected set of adjacent witness points.
- Transverse adjacency: Π_i can also cross Π_j transversely or simply intersect it at one point. In this case, the witness points $\boldsymbol{w}_n^i$ of Π_i that are both in primitive i and j form either one or two simply connected sets.

Note that in the presence of a primitive thinner than the tolerance, several sets of witnesses may appear. These cases are dealt with by the clustering phase (see Section 9.5.4).

Outer Corners

We define a notion of corner associated to a primitive. Although this notion is purely topological so far, it corresponds to the presence of a vertex in the meshing stage (see Section 9.6).

Considering a primitive Π_i, we generate corners by visiting the list of its witnesses $\boldsymbol{w}_n^i$. When two consecutive witnesses $\boldsymbol{w}_n^i$ and $\boldsymbol{w}_{n+1}^i$ are not tagged with the same set of primitives, a corner is inserted between them in the cyclic list of Π_i.

We tag this corner based on the primitives that are adjacent to only one of the two witness points that triggered its creation. The ordered sequence of corners placed along

the hull boundary will be connected by a set of constrained edges during meshing (see Section 9.6). Figure 9.7 (top right) depicts the corner generation.

Inner Corners

Generating corners on the boundaries of the primitives is not sufficient in the presence of non-manifold features.

To keep consistency between adjacencies, if a witness point of Π_i is in Π_j, we generate an inner corner in Π_j. Because of the convexity property, any segment connecting two of its inliers is entirely contained in the primitive. Two inner corners in Π_i linked to a common primitive are later connected by a constrained edge during meshing, while an isolated inner corner will be inserted as a vertex into the triangulation (see Section 9.6).

Boundary Corners

When the inferred shape contains open boundaries (edges that do not connect two primitives), an additional set of corners must be generated to properly approximate these boundaries. To this end, additional corners are greedily inserted between the witnesses $\boldsymbol{w}_n^i$ tagged with the corresponding primitive. More specifically, witnesses inserted one at a time until all witness points are well approximated by the set of boundary edges under the tolerance ε_S.

9.5.4 Embedding

In order to compute a 3D geometric vertex from the topological information of the corners previously computed, we define a cluster $\boldsymbol{C}_N$ that contains several corners $\boldsymbol{c}_n^i$ associated to different primitives.

Corner Clustering

First, a transitive closure of the adjacencies of corners is performed: the combinatorial information of the corners (their tags) is used to propagate them to other primitives.

Consider a corner $\boldsymbol{c}_n^i$ of Π_i that is adjacent to Π_j. We repeatedly merge this corner with the closest corner $\boldsymbol{c}_m^j$ of Π_j that is less than ε_S-away. These merged corners form clusters $\boldsymbol{C}_N$ which now represent the confluence of several primitives.

Because adjacencies with high degrees unnecessarily increase the complexity of the output, we also cluster the corners spatially: corners less than ε_S-away from each other are merged in the same cluster.

Once this clustering process is achieved, a unique topological vertex is generated per cluster of corners.

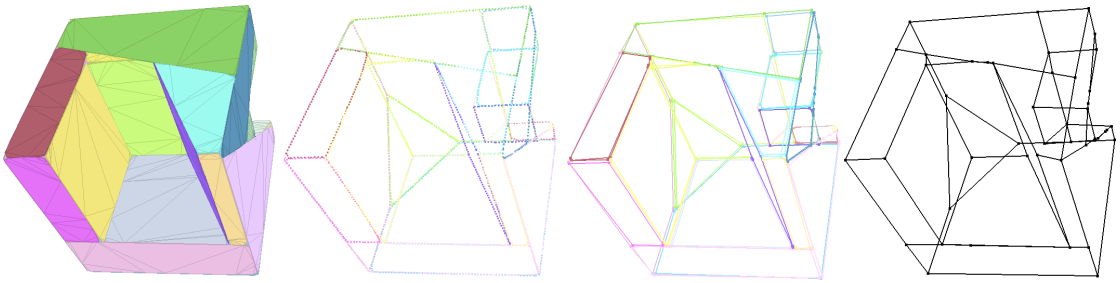


Figure 9.9: **Recovering adjacency on a generated point cloud.** Top: primitive set & witness points. Bottom: contouring of primitive & output after clustering and optimizing.

Embedding

Each primitive's boundary is defined by an ordered set of topological clusters $\boldsymbol{C}_N$. In order to get vertices defined in $\mathbb{R}^3$, we need to properly locate these clusters based on the geometrical information of their support primitives.

As a corner $\boldsymbol{c}_n^i$ is part of an ordered list of witness points, we can identify two adjacent edges on its primitive Π_i: suppose the next corner in the list is at position $n+\delta$, then the succeeding edge of $\boldsymbol{c}_n^i$ contains witnesses between $\boldsymbol{w}_{n+1}^i$ and $\boldsymbol{w}_{n+\delta-1}^i$. We compute the position of the line which minimizes the error to these witness points through principal component analysis, and proceed similarly for the preceding edge.

For a cluster $\boldsymbol{C}_N$ containing m corners, we therefore get a set of $2m$ support lines l_i. The position of the corresponding 3D vertex $\boldsymbol{v}$ is defined as the point that minimizes the distances to these $2m$ lines in the least squares sense: $\boldsymbol{v} = \arg\min_{\boldsymbol{v}} \sum_{i=1}^{2m} d(\boldsymbol{v}, l_i)^2$.

In some degenerate cases (lines close to parallel, for example), this least squares approach is not reliable, leading to strong distortions. When the vertex location is more than ε_S-away from all the corners of $\boldsymbol{C}_N$, we use a simple average of the corner points positions as the vertex location instead.

Figure 9.9 depicts an example of the adjacency algorithm.

9.6 Meshing

With the set of primitive $\{\Pi_i\}$ computed in Section 9.4 and their geometric adjacency relationships computed in Section 9.5, we now have all information required to generate a conforming mesh.

9.6.1 Facet Creation

The boundaries of each primitive Π_i is well-defined as an ordered set of corners $\boldsymbol{c}_n^i$ potentially embedded in clusters $\boldsymbol{C}_N$ shared with some other primitives.

In order to create well-shaped facets out of one primitive Π_i, we use the supporting plane S_i as a reference domain and compute a 2D *constrained* Delaunay triangulation of the primitive. Specifically, we project the corners $\boldsymbol{c}_n^i$ of Π_i on S_i and set a constraint between each pair of consecutive corners. This guarantees the preservation of edges and potential concavities created by the 3D embedding of vertices associated to the corner clusters.

Keeping the combinatorial information of vertices and facets from the 2D triangulation, we then map the Delaunay vertices back to their corresponding 3D vertices.

Each primitive is dealt with similarly while making sure that shared corners and shared edges are instantiated once in the final 3D surface mesh.

9.6.2 Non-Manifold Features

Handling non-manifold features is also made possible by the use of 2D constrained Delaunay triangulations. After building the associated constrained triangulation of a primitive Π_i, we identify the facets inside the boundary constraints. We can now set additional internal constraints that correspond to the non-manifold features. Specifically, we distinguish between four cases:

- **Isolated vertex:** primitive(s) adjacent to Π_i by a single isolated point located inside Π_i.
- **Internal edge:** primitive(s) adjacent to Π_i by an edge connecting 2 points located inside Π_i.

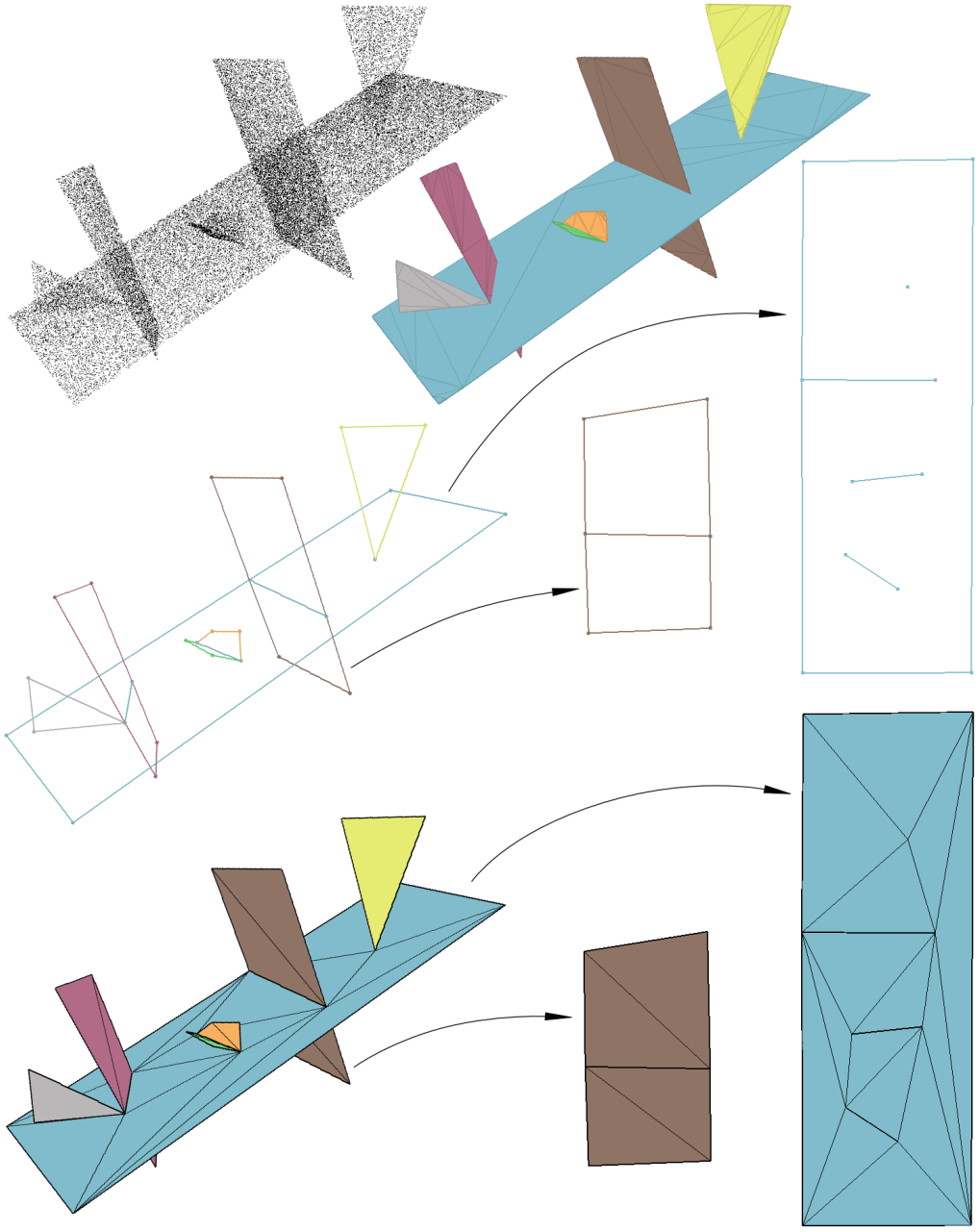


Figure 9.10: **Meshing of a non-manifold surface.** Top: input point cloud & primitive set. Middle: adjacencies with close-ups on two primitives. Bottom: conforming mesh with close-ups on two primitives.

- **Transverse edge:** primitive(s) adjacent to Π_i by an edge connecting 2 non-consecutive boundary vertices Π_i.
- **Semi-transverse edge:** primitive(s) adjacent to Π_i by an edge connecting a boundary vertex of Π_i to a point located inside Π_i.

Figure 9.10 depicts several non-manifold features.

Our output mesh is by construction a surface triangle mesh, but we can also visualized it as a polygon mesh, each primitive being mapped to a set of adjacent facets defining a close-to-planar polygon. We show our results either using colors to indicate the convex decomposition formed by the primitives, or through a simplicial complex where each edge is drawn for clarity.

9.7 Results

The algorithm is implemented in C++ using the *CGAL library* [1] and the *Eigen library* [42] for least squares fitting. All experiments are performed on an *Intel* 2.4GHz laptop with 8GB RAM.

Fig.	#Points	ε_N	ε_S	Time (s)	#Primitives	#Facets
9.5	$50K$	0.005	0.025	8	25	84
9.10	$50K$	0.005	0.015	7	7	27
9.12	$582K$	0.001	0.003	340	814	3,420
9.13	$2.0M$	0.002	0.002	1,872	3,488	7,533
9.14	$1.2M$	0.004	0.008	266	131	1,080
9.15	$1.3M$	0.001	0.002	523	625	2,706
9.17a	$224K$	0.018	0.010	48	84	329
9.17b	$1.2M$	0.008	0.004	435	1,569	3,003

Figure 9.11: **Timings.** Point clouds with number of points, timing in seconds, bounds ε_N and ε_S, number of primitives generated and number of triangle facets in the output mesh.

Timings and parameter settings of the presented reconstructions are detailed on Figure 9.11. Note that the total computation time depends more on the amount of details in the scene than on the total number of points. Most of the time (95%) is spent generating the primitives.

9.7.1 Checking Validity

We first perform some tests on simple generate data to estimate the validity of our algorithm. As expected, it performs well on a synthetic piecewise-planar surface as shown in Figure 9.5, producing a coarse and boundary-preserving triangle mesh.

One strength of our algorithm is its ability to conform even to non-manifold features, which is demonstrated by Figure 9.10.

The core targets of our algorithm are measurement data sets: piecewise-planar reconstruction is particularly well suited to man-made shapes such as urban scenes.

Figure 9.12 illustrates our algorithm at work on a dense LIDAR point cloud sampled on the outside of a building (582,582 points), using thresholds $\varepsilon_S = 0.001$ and $\varepsilon_N = 0.003$ (found automatically). The primitives generated during the reconstruction form a convex

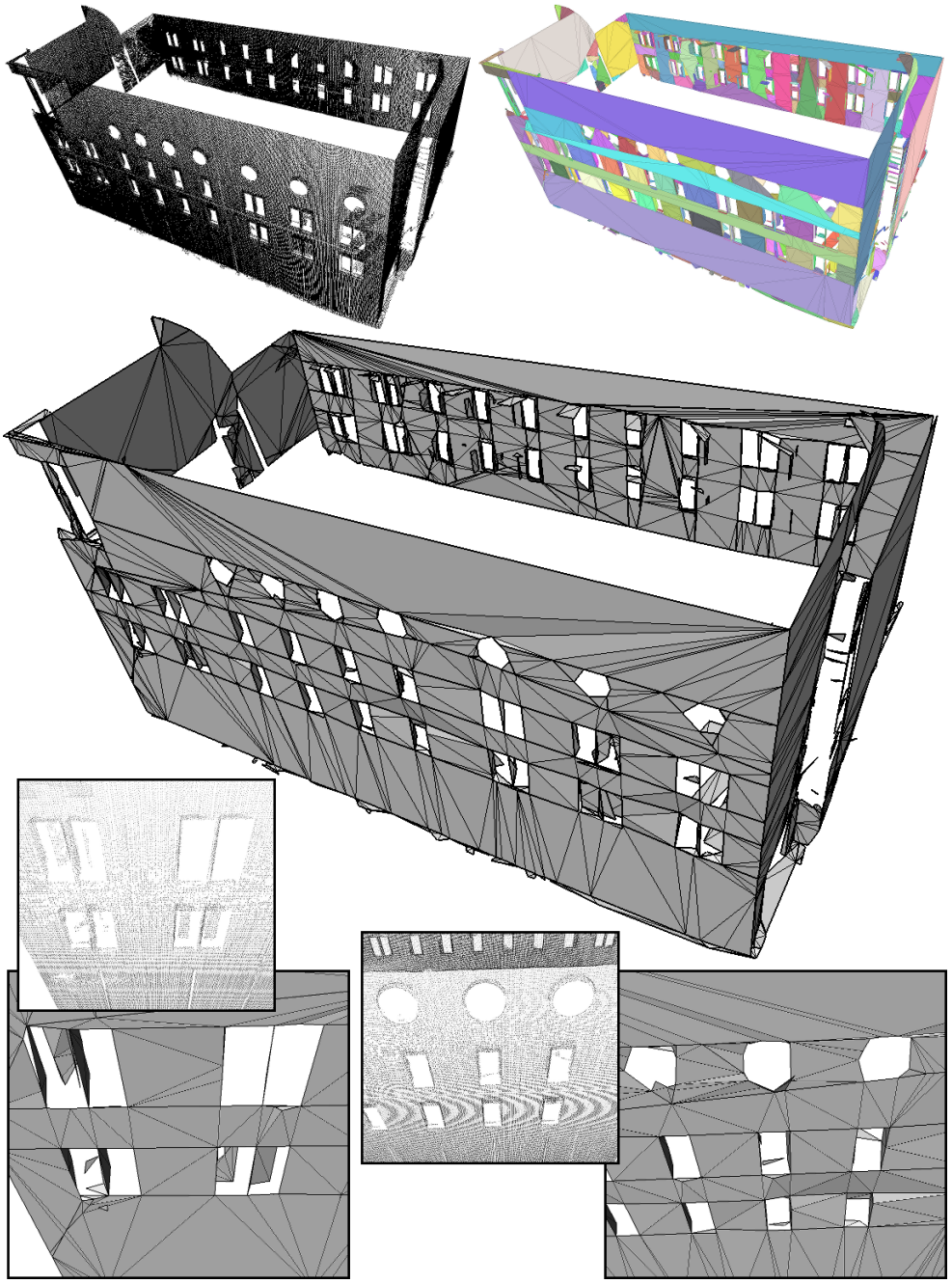


Figure 9.12: **Building.** Input LIDAR point cloud (top), colored primitives, output simplicial mesh, and two close-ups (bottom).

decomposition of the point cloud. The output mesh (3,420 triangle facets) is conforming to the facade and to the orthogonal window frames acquired by the LIDAR sensor. Note that it also adapts well to missing data.

Another example of a LIDAR point cloud is given in Figure 9.13. This one contains a wide range of details and large regions of missing data. When setting a small approximation parameter ε_S, the primitives locally adapt to the geometry of the fine details (roof tiles, small shutters). While it efficiently captures the neighboring large walls, the connection of widely different sized primitives yields skinny triangles.

9.7.2 Comparisons

As discussed in Section 6, literature in piecewise-planar reconstruction is vast and diverse. We compare our approach to several of these reconstruction methods in Figure 9.14, using a challenging example of a scanned church as input point cloud:

- Poisson reconstruction [48]: since this approach was devised to reconstruct smooth closed surfaces from point clouds with oriented normals, we simplify their output mesh via QEM-based decimation to obtain a mesh with the same complexity as ours.
- Scale space [32]: this approach generates a dense mesh that we also simplify to the same complexity as ours. Note that the fine details of the tiles hamper the feature-preserving decimation phase.
- Point cloud structuring [55]: this method is hybrid in the sense that it deals with both planar and free-form surfaces; we use the recommended parameters to generate as output a (mostly) piecewise-planar mesh. The dense mesh near boundaries is a consequence of the boundary preserving property of the algorithm (edges were not shown in the original paper).
- O-Snap algorithm [8]: this approach computes adjacency relations after performing RANSAC for planar primitive detection. We compare our results only to the automatic part of the algorithm (image taken from the original paper). As our approach is based on the Hausdorff distance to the input point cloud, our results tend to better preserve the details and boundaries (see circular roof and tower windows), while their approach excels at performing abstraction with a handful of planar polygons.

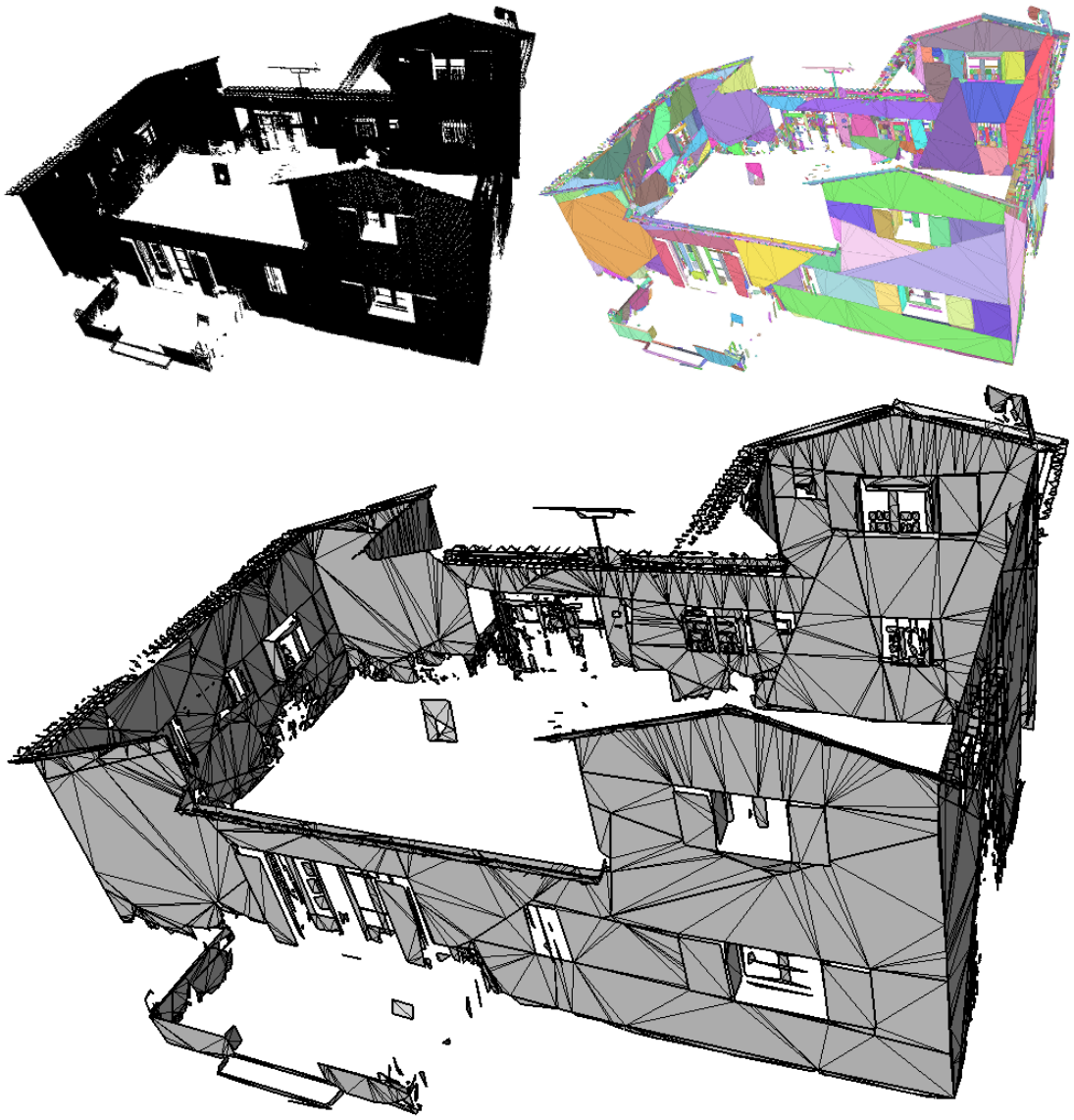


Figure 9.13: **House.** Input LIDAR point cloud ($2M$ points, top left), $1.8K$ colored primitives (top right), and output mesh ($3.4K$ facets, bottom).

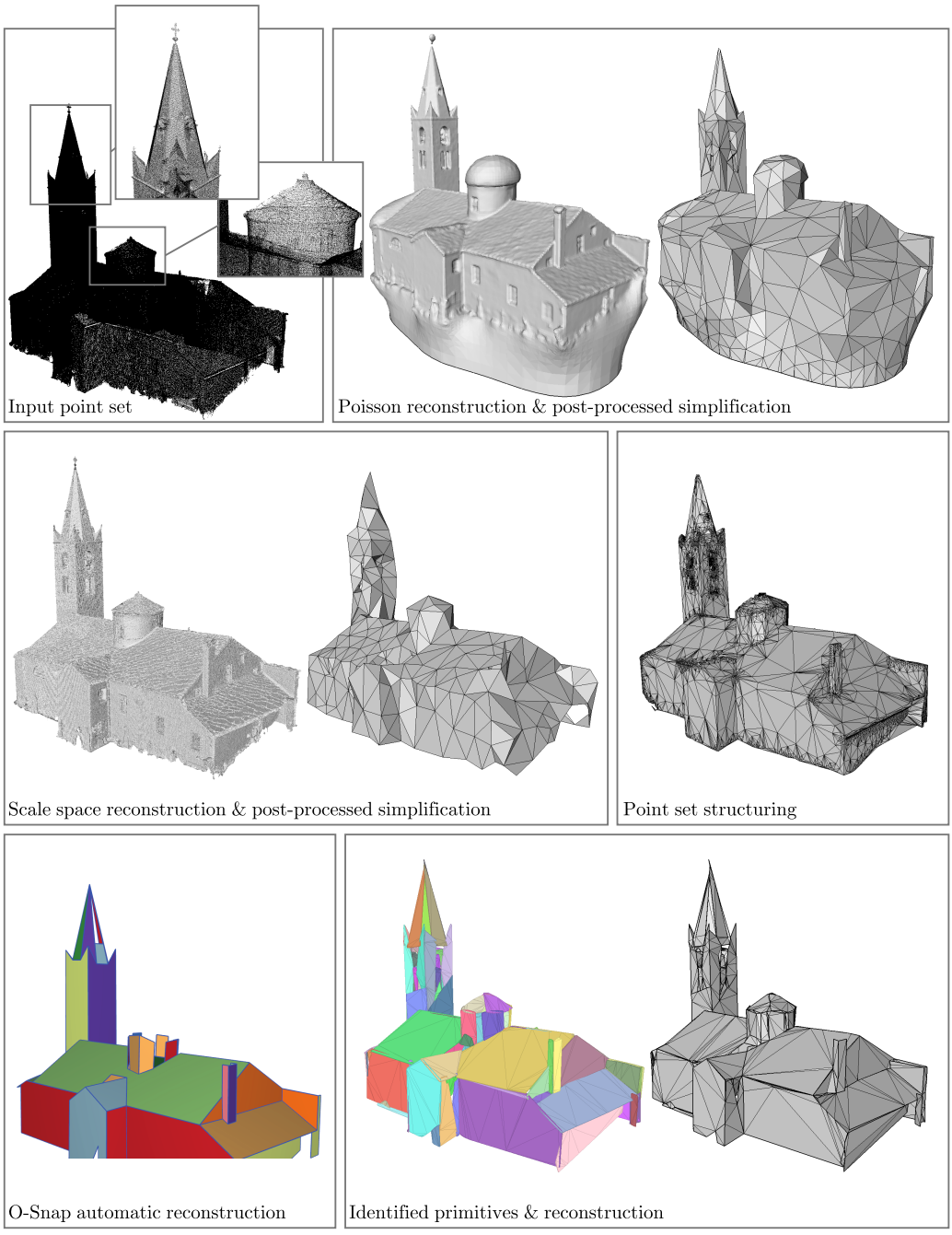


Figure 9.14: **Comparisons.** Input point cloud acquired on the church of Lans le Villar (1.2M points).

9.7.3 Robustness

We now challenge the robustness of our algorithm with complex point clouds.

We perform a stress test using an indoor low-quality LIDAR point cloud with noise and variable density, shown on Figure 9.15. The scene is a mixture of large planar surfaces, cluttered geometry, and complex boundaries due to many occlusions. Because the primitives are bound to approximate the surface with a sampling rate of ε_S, multiple connected components are generated due to the many invisible areas. Nevertheless, the algorithm is reliable, and each parts with sufficient sampling density is properly captured.

We then run our algorithm on different levels of noise until failure case (Figure 9.16). Robustness to noise is achieved through the single parameter ε_N from which a tight control over the Hausdorff distance between primitives and point cloud is enforced. This parameter is evaluated from the input data by analyzing the Hausdorff distances of primitives built from the most planar clusters of the coarsest scale. Note that our algorithm is not outlier robust: we cannot distinguish features from outliers.

Finally, Figure 9.17 shows the output of our algorithm on piecewise-smooth surfaces. On the mechanical part, the spherical cap is approximated by isotropic primitives and the mesh is watertight. The convexity and planarity properties of the primitives lead to a suboptimal meshing of the elliptic regions.

The last point cloud (*Bimba*) is smooth everywhere except on the bottom. Where the curvature of the inferred surface is high compared to the tolerance error (see left side of the hair), our algorithm fails in recovering a surface as the local planarity assumption is not met locally.

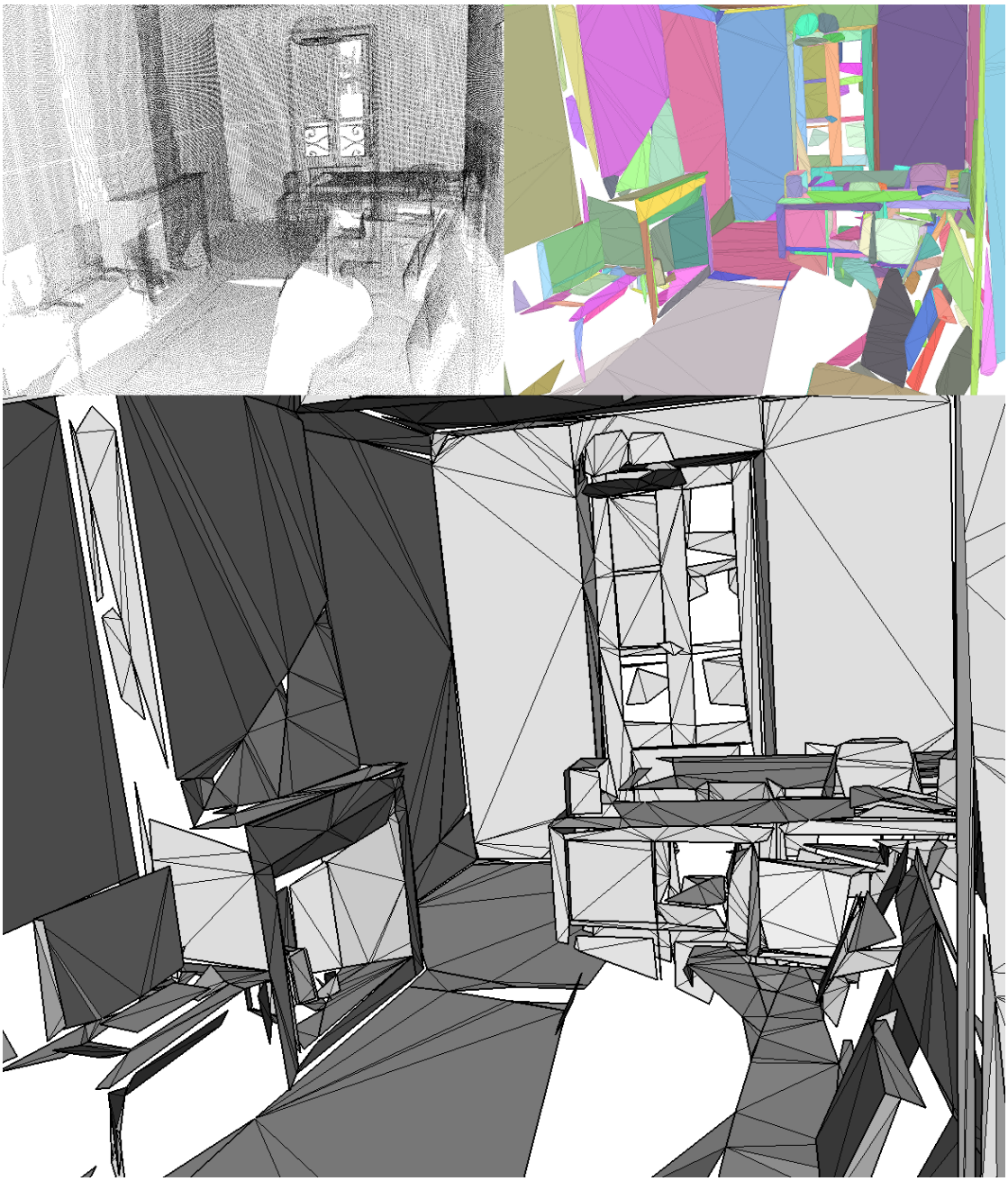


Figure 9.15: **Indoor scene.** Input LIDAR point cloud (1.3M points), 625 colored primitives, and the output mesh (2.7K facets).

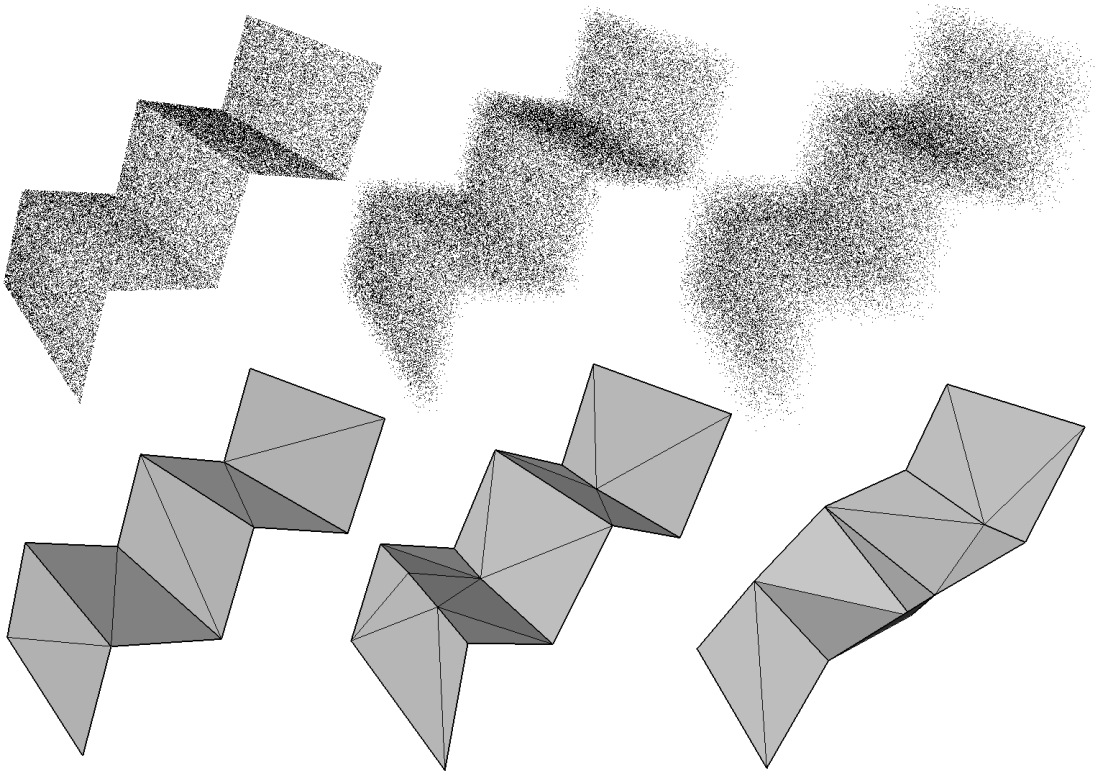


Figure 9.16: **Noise robustness.** Top: input point cloud with increasing level of noise. Bottom: output mesh using a noise threshold ε_N of respectively 0.005, 0.050 and 0.200.

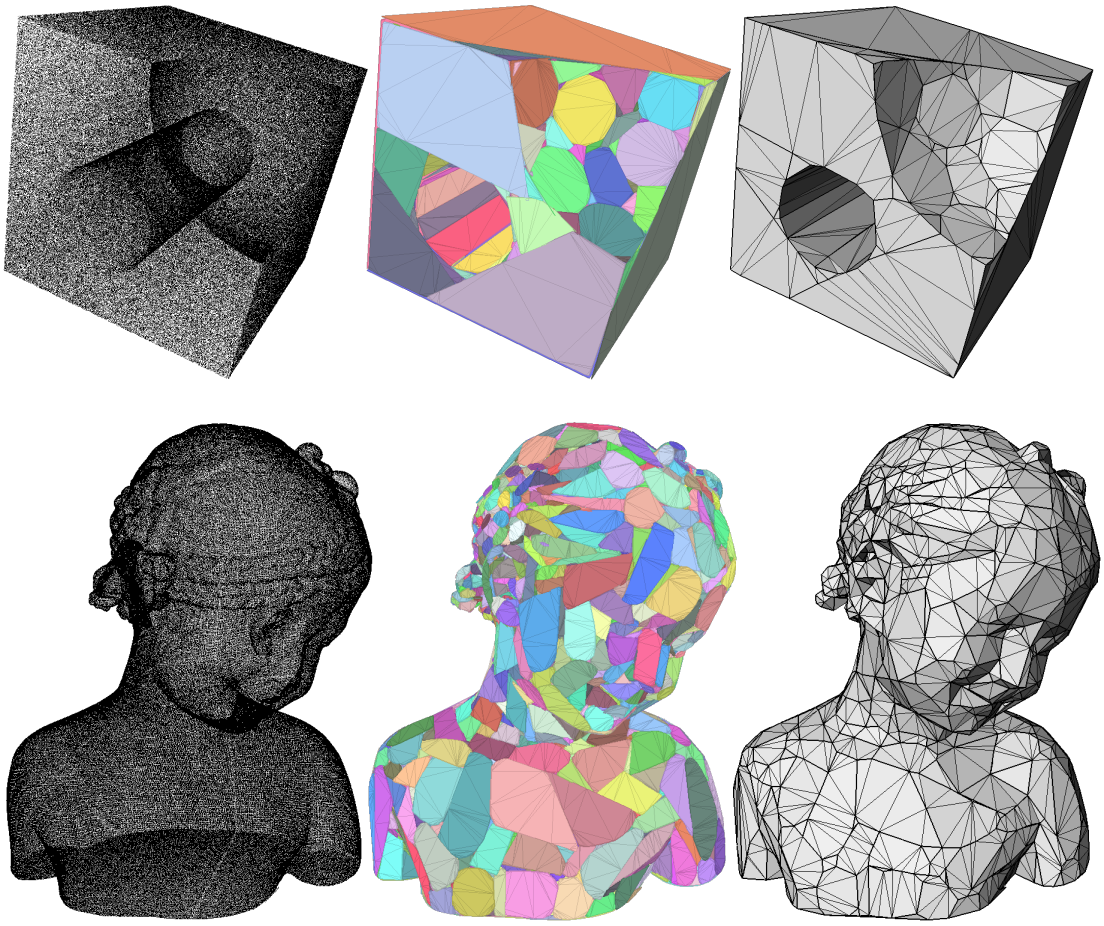


Figure 9.17: **Piecewise-smooth cases.** Input point clouds, primitives & output meshes.

9.8 Conclusions

We presented a novel approach to perform piecewise-planar reconstruction from raw point clouds. It uses convex, planar primitives with bounded Hausdorff error with respect to the input. An adjacency graph between the primitives is computed by leveraging the convexity of the primitives. A coarse mesh is finally extracted, reliably inferring the geometry of the shape.

9.8.1 Contribution

The main strength of our algorithm is the preservation of boundaries and sharp features. Even in the cases of non-manifold features, we generate coarse conforming meshes.

In addition, our approach is also reliable and guaranteed to terminate. It relies only on simple and mature algorithms from computational geometry: point cloud clustering, 2D convex hull, 2D Delaunay triangulation and 3D Delaunay tetrahedralization.

Finally, our implementation scales well, in particular our multiscale framework yields a computational complexity that depends more on the total amount of details in the scene than on the cardinality of the input point cloud.

9.8.2 Limitations

Our approach aims at faithfully approximating the input point cloud with a strong requirement on the sampling density. This becomes a limitation in the sense that no data completion can be performed. Instead, missing data is an area with an additional set of sharp boundaries reconstructed.

Although the embedding of the vertices of the output mesh is optimized in a least squares fashion, the positions of the edges are only enforced topologically depending on their end point vertices. Edge locations could also be optimized with respect to their adjacent primitives planes.

III

Conclusion

We have contributed two novel shape reconstruction approaches designed for two specific classes of output. Both algorithms take raw point clouds as input, possibly ridden with defects such as noise and outliers.

The first approach focuses on smooth, closed shapes and addresses the problem of robustness to variable noise. The second approach reconstructs piecewise-planar surfaces through an efficient convex decomposition with global error bounds.

Both approaches mainly rely on a dimension assumption. More specifically, the reconstruction should match the one of the inferred shape, a curve in 2D or a surface in 3D. For the first approach, the dimension assumption is used via computing a noise-adaptive distance function that selects the smallest scale that matches the dimension of the inferred shape. For the second approach, the dimension assumption triggered the construction of planar, convex primitives that match the dimension of the inferred surface with bounds on the two-sided Hausdorff distance.

Robustness

Our main focus is the robustness to various defects.

Robustness to noise is achieved through two robust metrics. First, the noise-adaptive distance function is robust to both outliers and variable levels of noise. Furthermore, the isolevels of this function are converging uniformly to the inferred shape. Second, the two-sided Hausdorff distance provides global error bounds between the set of convex primitives and the input point cloud, and is robust to noise with lower magnitude than the tolerance error.

Robustness to missing data is dealt with in two different ways. For smooth closed surfaces, data completion is achieved through signing the robust distance function. We consolidate sign guesses through local and non-local hypotheses represented on a random graph. For piecewise-planar surfaces, no data completion is performed and hence missing data translate into a larger number of boundaries.

Outlook

Albeit this thesis focuses on the shape reconstruction problem, we touched other related scientific challenges. We now provide an outlook on some of these challenges.

Multiscale reconstruction

As discussed in Part I, the noise-adaptive distance function not only localizes the inferred shape, but also provides knowledge on higher scales where, for instance, two nearby surface sheets appear as only one noisy surface. Our choice is to rely upon the assumption that the finest scale should be reconstructed in order to faithfully recover the details of the inferred shape. However, several surfaces can be reconstructed when considering the multiple scales of a scene. Such multi-scale viewpoint is relevant for scenes comprising a wide range of levels of details. In future work we plan to investigate a hierarchical or a progressive reconstruction approach via scale decomposition.

Compression

Our approach to convex decomposition of point clouds (Part II) may be seen as a lossy data compression approach for point clouds. A sparse set of primitives is sufficient to reconstruct massive point clouds, and each convex primitive is represented by a bounded number of extreme points through discretization of the orientations. The total number of extreme points is substantially lower than the total number of input points. As future work we plan to devise an approach for encoding these extreme points, progressive or single-rate, which may be seen as one step toward effective point cloud compression algorithm.

Global error bounds

Our piecewise-planar reconstruction approach (Part II) starts with a convex decomposition. It relies on the reliably of the Hausdorff distance to faithfully approximate the input point cloud with global error bounds. However, when the adjacencies are computed and the mesh is generated, the validity of the global error bound may be violated. Although our embedding algorithm for vertex placement ensures that the distortion of the corners of the primitives does not exceed the global error bounds, we cannot guarantee that these

bounds are not violated by the facets. In future work we plan to preserve these global error bounds even after meshing. One direction is to constrain all facets of each primitive to be strictly coplanar, but this substantially increases the complexity of the output mesh.

Conclusion (français)

Notre contribution se compose de deux nouvelles approches de reconstruction de formes dédiées à deux catégories spécifiques de données. Chaque algorithme prend un nuage de points brut en entrée, potentiellement accompagné de défauts tels que du bruit ou des données aberrantes.

La première approche se concentre sur les formes lisses fermées et répond au problème de la robustesse au bruit variable. La seconde approche reconstruit des surfaces planes par morceaux en utilisant une décomposition convexe efficace avec des bornes d'erreur globales.

Chacune des deux approches repose principalement sur une hypothèse de dimension. Plus précisément, la reconstruction doit satisfaire la dimension de la forme recherchée, une courbe en 2D ou une surface en 3D. Pour la première approche, cette hypothèse de dimension est utilisée en calculant une fonction distance adaptative au bruit qui sélectionne la plus petite échelle qui correspond à la dimension de la forme recherchée. Pour la seconde approche, l'hypothèse de dimension motive la construction de primitives convexes et planes qui approchent la surface recherchée avec des bornes sur la distance de Hausdorff symmétrique.

Robustesse

Le cœur de notre problème est la robustesse à divers défauts.

La robustesse au bruit est atteinte à travers deux métriques robustes. Premièrement, la fonction distance adaptative au bruit est robuste à la fois aux données aberrantes et à des niveaux variables de bruit. De plus, les iso-niveaux de cette fonction convergent uniformément vers la forme recherchée. Deuxièmement, la distance de Hausdorff symmétrique procure des bornes d'erreur globales entre l'ensemble de primitives convexes et le nuage

de points d'entrée, et elle est robuste au bruit de magnitude plus faible que la tolérance d'erreur.

La robustesse aux données manquantes est traitée de deux manières différentes. Pour les surfaces lisses fermées, signer la fonction distance robuste permet de compléter les données. Les signes supposés sont consolidés par des hypothèses à la fois locales et non-locales représentées sur un graphe aléatoire. Pour les surfaces planes par morceaux, les données ne sont pas complétées et, de fait, des données manquantes se traduisent par un plus grand nombre de bords.

Perspectives

Bien que cette thèse se concentre sur le problème de la reconstruction de formes, nous avons effleuré d'autres défi scientifiques liés. Nous présentons maintenant des perspectives sur certains de ces défis.

Reconstruction multi-échelle

Comme cela est évoqué dans la Partie I, la fonction distance adaptative au bruit localise non seulement la forme recherchée, mais elle offre aussi des informations sur les échelles plus larges où, par exemple, deux feuilles de la surface proches apparaissent comme une unique surface bruitée. Notre choix est de reposer sur l'hypothèse que l'échelle la plus fine est celle à reconstruire pour retrouver les détails de la forme recherchée avec fidélité. Cependant, plusieurs surfaces peuvent être reconstruites si l'on considère les multiples échelles d'une scène. Ce point de vue multi-échelle a du sens pour les scènes qui comprennent un large panel de niveaux de détails. Dans de prochains travaux, nous avons l'intention d'étudier des approches de reconstruction hiérarchiques ou progressives à travers cette décomposition des échelles.

Compression

Notre approche de décomposition convexe de nuages de points (Partie II) peut être vue comme une méthode de compression de données pour des nuages de points. Un petit ensemble de primitives est suffisant pour reconstruire d'immenses nuages de points et chaque

primitive convexe est représentée par un nombre borné de points extrêmes calculés par discrétisation des orientations. Le nombre total de points extrêmes est considérablement plus faible que le nombre total de points de donnée. Dans de prochains travaux, nous avons l'intention de concevoir une méthode pour encoder ces points extrêmes de manière progressive ou à taux constant, ce qui peut être vu comme un premier pas vers un algorithme efficace de compression de nuages de points.

Bornes d'erreur globales

Notre méthode de reconstruction plane par morceaux (Partie II) se base sur une décomposition convexe. Elle repose sur la fiabilité de la distance de Hausdorff pour approcher de manière fidèle le nuage de points d'entrée avec des bornes d'erreur globales. Cependant, lors du calcul des adjacences et lors de la génération de maillage, ces bornes d'erreur globales peuvent être dépassées. Bien que notre algorithme de placement des sommets s'assure que la distorsion des coins des primitives ne dépasse pas les bornes d'erreur globales, nous ne pouvons garantir que ces bornes sont encore respectées sur les facettes. Dans de futurs travaux, nous comptons préserver ces bornes d'erreur globales jusqu'à la phase de maillage. Une solution serait de contraindre les facettes de chaque primitive à rester strictement coplanaires, mais cela augmenterait considérablement la complexité du maillage de sortie.

List of Figures

Index

Bibliography

[1] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[2] *EPFL Computer Graphics and Geometry Laboratory*, EPFL statue model repository. http://lgg.epfl.ch/statues.php?p=dataset, 2012.

[3] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Point set surfaces. In *Proceedings of the Conference on Visualization '01*, VIS '01, pages 21–28, Washington, DC, USA, 2001. IEEE Computer Society.

[4] Pierre Alliez, Simon Giraudot, and David Cohen-Steiner. Robust shape reconstruction and optimal transportation. In *Courbure discrète: théorie et applications*, volume 3, 2014.

[5] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, SCG '98, pages 39–48, New York, NY, USA, 1998. ACM.

[6] Nina Amenta, Marshall Bern, and David Eppstein. The crust and the beta-skeleton: Combinatorial curve reconstruction. In *Graphical Models and Image Processing*, pages 125–135, 1998.

[7] Nina Amenta and Yong Kil. The domain of a point set surface. In *Proceedings of EUROGRAPHICS conference on Point-Based Graphics*, pages 139–147, 2004.

[8] Murat Arikan, Michael Schwärzler, Simon Flöry, Michael Wimmer, and Stefan Maierhofer. O-snap: Optimization-based snapping for modeling architecture. *ACM Transactions on Graphics*, 32:6:1–6:15, January 2013.

[9] Dominique Attali. r-regular shape reconstruction from unorganized points. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 248–253. ACM, 1997.

[10] Marco Attene, Michela Mortara, Michela Spagnuolo, and Bianca Falcidieno. Hierarchical convex approximation of 3d shapes for fast region selection. In *Computer graphics forum*, volume 27, pages 1323–1332. Wiley Online Library, 2008.

[11] Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. L1-sparse reconstruction of sharp point set surfaces. *ACM Transactions on Graphics*, 29:135:1–135:12, November 2010.

[12] Matthew Berger and Claudio Silva. Medial kernels. *Computer Graphics Forum*, 31:795–804, 2012.

[13] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, October 1999.

[14] Matthew Bolitho, Michael M. Kazhdan, Randal C. Burns, and Hugues Hoppe. Multilevel streaming for out-of-core surface reconstruction. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Barcelona, Spain, July 4-6, 2007*, pages 69–78, 2007.

[15] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nüchter. The 3D Hough Transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2(2), 2011.

[16] Alexandre Boulch, Martin de La Gorce, and Renaud Marlet. Piecewise-planar 3D reconstruction with edge and corner regularization. In *Computer Graphics Forum*, volume 33(5), pages 55–64, 2014.

[17] Alexandre Boulch and Renaud Marlet. Fast and robust normal estimation for point clouds with sharp features. *Comp. Graph. Forum*, 31(5):1765–1774, August 2012.

[18] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000.

[19] Ricard Campos Dausà et al. *Surface reconstruction methods for seafloor modelling.* PhD thesis, 2014.

[20] E. J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theor.*, 52(2):489–509, February 2006.

[21] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 67–76, New York, NY, USA, 2001. ACM.

[22] A-L Chauve, Patrick Labatut, and J-P Pons. Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *Computer Vision and Pattern Recognition*, pages 1261–1268, 2010.

[23] Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric inference for probability measures. *Foundations of Computational Mathematics*, 11(6):733–751, 2011.

[24] Xuan Cheng, Ming Zeng, and Xinguo Liu. Feature-preserving filtering with l_0 gradient minimization. *Computers & Graphics*, 38:150–157, 2014.

[25] Joel Daniels, Linh K. Ha, Tilo Ochotta, and Claudio T. Silva. Robust smooth feature extraction from point clouds. In *Shape Modeling and Applications*, pages 123–136, 2007.

[26] Fernando de Goes, David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. An optimal transport approach to robust reconstruction and simplification of 2d shapes. *Computer Graphics Forum*, 30(5):1593–1602, 2011. Proceedings of EUROGRAPHICS Symposium on Geometry Processing.

[27] Kris Demarsin, Denis Vanderstraeten, Tim Volodine, and Dirk Roose. Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Computer-Aided Design*, 39(4):276–283, 2007.

[28] Tamal K. Dey, Xiaoyin Ge, Qichao Que, Issam Safa, L. Wang, and Yusu Wang. Feature-preserving reconstruction of singular surfaces. *Computer Graphics Forum*, 31(5):1787–1796, 2012.

[29] Tamal K. Dey and Samrat Goswami. Provable surface reconstruction from noisy samples. *Comput. Geom. Theory Appl.*, 35(1):124–141, August 2006.

[30] Julie Digne. *Inverse Geometry: From the raw point cloud to the 3D Surface (Theory and Algorithms)*. PhD thesis, Ecole normale superieure de Cachan, 11 2010.

[31] Julie Digne, David Cohen-Steiner, Pierre Alliez, Fernando De Goes, and Mathieu Desbrun. Feature-Preserving Surface Reconstruction and Simplification from Defect-Laden Point Sets. *Journal of Mathematical Imaging and Vision*, pages 1–14, January 2013.

[32] Julie Digne, Jean-Michel Morel, Charyar-Mehdi Souzani, and Claire Lartigue. Scale space meshing of raw data point sets. *Computer Graphics Forum*, pages 1630–1642, 2011.

[33] D.L. Donoho, M. Elad, and V.N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *Information Theory, IEEE Transactions on*, 52(1):6–18, Jan 2006.

[34] Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 13(1):43–72, January 1994.

[35] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[36] Shachar Fleishman, Daniel Cohen-Or, Marc Alexa, and Cláudio T. Silva. Progressive point set surfaces. *ACM Trans. Graph.*, 22(4):997–1011, October 2003.

[37] Simon Giraudot, David Cohen-Steiner, and Pierre Alliez. Noise-adaptive shape reconstruction from raw point sets. In *Computer Graphics Forum*, volume 32, pages 229–238. Wiley Online Library, 2013.

[38] David Gisch and Jason M Ribando. Apollonius' problem: A study of solutions and their connections. *American Journal of Undergraduate Research*, 3(1):15–25, 2004.

[39] Nuno Gracias, Pere Ridao, Rafael Garcia, J Escartin, Michel L'Hour, Franca Cibecchini, Ricard Campos, Marc Carreras, David Ribas, Narcıs Palomeras, et al. Mapping

the moon: Using a lightweight auv to survey the site of the 17th century ship 'la lune'. In *OCEANS-Bergen, 2013 MTS/IEEE*, pages 1–8. IEEE, 2013.

[40] Leo Grady. Random walks for image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(11):1768–1783, 2006.

[41] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 26(3), July 2007.

[42] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[43] Stefan Gumhold, Xinlong Wang, and Robert MacLeod. Feature extraction from point clouds. In *International Meshing Roundtable*, pages 293–305, October 2001.

[44] Lei He and Scott Schaefer. Mesh denoising via l_0 minimization. *ACM Trans. Graph*, 32(4):64, 2013.

[45] Dirk Holz and Sven Behnke. Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. In *Intelligent Autonomous Systems 12*, volume 194, pages 61–73, 2012.

[46] Philipp Jenke, Bastian Krückeberg, and Wolfgang Straßer. Surface reconstruction from fitted shape primitives. In *Vision, Modeling, and Visualization*, pages 31–40, 2008.

[47] Philipp Jenke, Michael Wand, and Wolfgang Straßer. Patch-graph reconstruction for piecewise smooth surfaces. In *Vision, Modeling, and Visualization*, pages 3–12, 2008.

[48] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of EUROGRAPHICS Symposium on Geometry Processing*, pages 61–70, 2006.

[49] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3):29:1–29:13, July 2013.

[50] Qifa Ke and Takeo Kanade. Robust l 1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 739–746. IEEE, 2005.

[51] Yong Se Kim. Recognition of form features using convex decomposition. *Computer-Aided Design*, 24(9):461–476, 1992.

[52] Ravikrishna Kolluri. Provably good moving least squares. *ACM Trans. Algorithms*, 4(2):18:1–18:25, May 2008.

[53] Nojun Kwak. Principal component analysis based on l1-norm maximization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(9):1672–1680, 2008.

[54] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Robust and efficient surface reconstruction from range data. *Computer Graphics Forum*, 28(8):2275–2290, 2009.

[55] Florent Lafarge and Pierre Alliez. Surface reconstruction through point set structuring. *Computer Graphics Forum*, 2013.

[56] David Levin. The approximation power of moving least-squares. *Math. Comput.*, 67(224):1517–1531, October 1998.

[57] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 30(4):52:1–52:12, July 2011.

[58] Yaron Lipman, Daniel Cohen-Or, and David Levin. Data-dependent mls for faithful surface approximation. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 59–67. Eurographics Association, 2007.

[59] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 26(3), July 2007.

[60] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, August 1987.

[61] Nicolas Mellado, Pascal Barla, Gaël Guennebaud, Patrick Reuter, and Christophe Schlick. Growing Least Squares for the Continuous Analysis of Manifolds in Scale-Space. *Computer Graphics Forum*, July 2012.

[62] Deyu Meng, Qian Zhao, and Zongben Xu. Improve robustness of sparse pca by l 1-norm maximization. *Pattern Recognition*, 45(1):487–497, 2012.

[63] Quentin Mérigot, Maks Ovsjanikov, and Leonidas J. Guibas. Voronoi-based curvature and feature estimation from point clouds. *IEEE Trans. Vis. Comput. Graph*, 17(6(17)):743–756, 2011.

[64] Patrick Mullen, Fernando De Goes, Mathieu Desbrun, David Cohen-Steiner, and Pierre Alliez. Signing the unsigned: Robust surface reconstruction from raw pointsets. *Computer Graphics Forum*, 29(5):1733–1741, 2010. Proceedings of EUROGRAPHICS Symposium on Geometry Processing.

[65] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. In *ACM SIGGRAPH 2005 Courses*, page 173. ACM, 2005.

[66] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. 3d scattered data approximation with adaptive compactly supported radial basis functions. In *Proceedings of Shape Modeling Applications*, pages 31–39, 2004.

[67] A. C. Öztireli, G. Guennebaud, and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum*, 28(2):493–501, 2009.

[68] Mark Pauly, Markus Gross, and Leif Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of IEEE Visualization*, pages 163–170, 2002.

[69] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, 22(3):281–290, 2003.

[70] T Rabbani, F van Den Heuvel, and G Vosselman. Segmentation of point clouds using smoothness constraint. In *"ISPRS Image Engineering and Vision Metrology"*, volume 36(5), 2006.

[71] Laurent Rineau and Mariette Yvinec. A generic software design for Delaunay refinement meshing. *Computational Geometry*, pages 100–110, 2007.

[72] Nader Salman, Mariette Yvinec, and Quentin Mérigot. Feature Preserving Mesh Generation from 3D Point Clouds. In *Computer Graphics Forum*, volume 29, pages 1623–1632, 2010.

[73] Ruwen Schnabel, Patrick Degener, and Reinhard Klein. Completion and reconstruction with primitive shapes. In *Computer Graphics Forum*, volume 28(2), pages 503–512. Wiley Online Library, 2009.

[74] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.

[75] Shy Shalom, Ariel Shamir, Hao Zhang, and Daniel Cohen-Or. Cone carving for surface reconstruction. *ACM Transactions on Graphics*, 29(6):150:1–150:10, December 2010.

[76] Kaleem Siddiqi and Stephen Pizer. *Medial Representations: Mathematics, Algorithms and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[77] Yuqing Song. Boundary fitting for 2D curve reconstruction. *The Visual Computer*, 26:187–204, 2010.

[78] S. Sotoodeh. Outlier detection in laser scanner point clouds. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5):297–302, 2006.

[79] Andrea Tagliasacchi, Matt Olson, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. Vase: Volume-aware surface evolution for surface reconstruction from incomplete point clouds. *Computer Graphics Forum*, 30(5):1563–1571, 2011.

[80] Ranjith Unnikrishnan, Jean-Francois Lalonde, Nicolas Vandapel, and Martial Hebert. Scale selection for geometric fitting in noisy point clouds. *International Journal on Computational Geometry and Applications*, 20(5), October 2010.

[81] Marc van Kreveld, Thijs van Lankveld, and Remco C Veltkamp. Watertight scenes from urban lidar and planar surfaces. *Symposium on Geometry Processing*, 2013.

[82] Hao Wang, Carlos Eduardo Scheidegger, and Claudio Silva. Bandwidth selection and reconstruction quality in point-based surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 15(4):572–582, 2009.

[83] Kun Zhou, Minmin Gong, Xin Huang, and Baining Guo. Data-parallel octrees for surface reconstruction. *Visualization and Computer Graphics, IEEE Transactions on*, 17(5):669–681, May 2011.

[84] Qian-Yi Zhou and Ulrich Neumann. 2.5D building modeling by discovering global regularities. In *Computer Vision and Pattern Recognition*, 2012.