Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Aproximación Eficiente de la Cápsula No-Convexa para Reconstrucción de Superficies

Tesis de Licenciatura en Ciencias de la Computación

Julián Bayardo Spadafora

Director: Francisco Gómez Fernández

Buenos Aires, 2019

APROXIMACIÓN EFICIENTE DE LA CÁPSULA NO-CONVEXA PARA RECONSTRUCCIÓN DE SUPERFICIES

La reconstrucción de superficies en tres dimensiones suele comenzar con una nube de puntos. Existen una multiplicidad de algoritmos utilizados para la reconstrucción, según las distintas suposiciones que se puedan hacer sobre la nube (el tipo de objeto que representa, o la metodología utilizada para obtenerla).

En este trabajo hacemos foco en un algoritmo denominado Naive Non-Convex Hull (Cápsula No-Convexa Ingenua), que reconstruye superficies utilizando un concepto similar a la Transformación del Eje Medial. Explicamos teoría a partir de la cual se llega al concepto de Cápsula No-Convexa, demostramos múltiples propiedades con respecto a la misma, y establecemos vínculos con otros conceptos utilizados en reconstrucción de superficies 3D, como el Power Diagram (Diagrama de Laguerre-Voronoi) y la Transformación del Eje Medial.

Basándonos en métodos de la literatura sobre la Transformación del Eje Medial, creamos uno nuevo llamado Contracción de Planos (Shrinking Planes, SP) para la Cápsula No-Convexa, corrigiendo en el proceso problemas de los métodos de referencia. Sobre el mismo demostramos propiedades de aproximación sin error, y evaluamos su capacidad de reconstrucción rigurosamente a través de múltiples experimentos cuantitativos y cualitativos. El nuevo método, además de lograr mantener la misma calidad de reconstrucción que el método original, logra hacerlo con una marcada mejora en su velocidad de ejecución.

Palabras claves: Reconstrucción de Superficies 3D, Transformación del Eje Medial, Geometría Constructiva Sólida, Superficies Implícitas

FAST NON-CONVEX HULL APPROXIMATION FOR SURFACE RECONSTRUCTION

3D Surface Reconstruction usually begins with a point cloud. There are several algorithms to solve this problem, each one with different priors over the point cloud (such as the kind of object represented, or the method by which it was obtained).

In this work, we focus on an algorithm called Naïve Non-Convex Hull, which reconstructs surfaces through a concept similar to the Medial Axis Transform. We explain the theory required to understand the Non-Convex Hull, prove several of its properties, and establish and explain links to other popular concepts in surface reconstruction, such as the Power Diagram and the Medial Axis Transform.

A new algorithm is proposed to compute the NCH, based on the Shrinking Ball method by Ma et al. [MBC12] with various improvements. We prove that the new method can approximate surfaces to arbitrarily small error, and rigorously evaluate its performance on the surface reconstruction task. The new method maintains the same reconstruction quality as the Naïve Non-Convex Hull method, while achieving a large performance improvement.

Keywords: 3D Surface Reconstruction, Medial Axis Transform, Constructive Solid Geometry, Implicit Surfaces, Shrinking Ball

AGRADECIMIENTOS

Este reporte es el resultado de una tesis realizada en el Grupo de Procesamiento de Imágenes y Visión por Computadora en la Facultad de Ciencias Exactas y Naturales, parte de la Universidad de Buenos Aires. Por este resultado, quiero agradecer a mi director, Francisco Gomez Fernandez, su conocimiento y toda la ayuda ofrecida en este proceso fueron fundamentales. Además, quiero agradecer a Gabriel Taubin por ofrecer su conocimiento y ayudarnos a concretar las ideas.

Quiero también agradecer a mis todos mis amigos, por haber estado en todo momento. Más importante aún, quiero agradecer a mi familia, a Lucía, y a Florencia, por siempre alentarme en todo lo que me propuse, y por ser mi soporte sentimental a lo largo de los años de carrera.

A mi vieja.

CONTENTS

1. INTRODUCTION

3D objects are usually represented as a polygon mesh. This representation is chosen because it is very efficient for most common operations. In 3D scanning, it is usual to obtain a unstructured point cloud instead, which is just a finite set of points $\mathcal{P} = \{\mathbf{p_1}, \ldots, \mathbf{p_n}\} \subset \mathbb{R}^3$ that lie in the boundary $\partial\mathcal{S}$ of the surface $\mathcal{S}$, with peculiar characteristics that come from the specific scanning methodology and the object being scanned. The point cloud is then processed into a polygon mesh, which can be used in any computer graphics application afterwards. This step is called 3D surface reconstruction, and it is what we are going to be concerned within this thesis.

As will be evident later on, reconstruction is a hard problem, with many caveats and border cases. Its many complexities lead algorithms to be rather hard to understand, code and parallelize.

The Medial Axis Transform (MAT) is a shape representation due to Blum [Blu67], in which shapes are given as a union of balls contained within it, each one called Medial Atom (MA). Since its invention, it has been used for surface reconstruction [ACK01], shape simplification [TH03] [BS12], point cloud simplification [MBC12] [Pet14d], skeletonization [MBC12] [JKT13], among other applications.

Taubin [Tau13] presented an algorithm, Non-Convex Hull (NCH) Surface Reconstruction, that is very simple to understand, code and parallelize, albeit slow for large point clouds. The algorithm is based on the concept of the NCH, which extends the MAT to allow planes, these can be seen as spheres of infinite radius. The extension seems simple, but is a crucial modification for the purpose of surface reconstruction, as will be shown later on.

His contribution went largely unnoticed, and there have not been any improvements to the method since. In this thesis, the intent is to analyze and improve the algorithm in order to make it applicable to large point clouds, as well provide the first step towards a state-of-the-art NCH-based surface reconstruction method.

1.1 Related Work

The approach taken in this thesis spans several different areas, such as Surface Reconstruction and Isoextraction, each of these is very extensive and thus we will discuss only the most relevant work. For the avid reader, citations to surveys will be given when

appropriate.

1.1.1 Surface Reconstruction

A vast amount of work has been dedicated to surface reconstruction over the past two decades. For a more complete literature review, Berger et al. [Ber+17] wrote a great survey, and Dey [Dey06] is a great book on the topic, covering many popular algorithms.

Hoppe et al. [Hop+92] wrote one of the first papers in the field. Techniques such as Poisson Surface Reconstruction by Kazhdan et al. [KBH06], Multi-Level Partition Of Unity by Ohtake et al. [Oht+03], Point-Set Surfaces by Alexa et al. [Ale+01], Algebraic Point-Set Surfaces by Guennebaud and Gross [GG07], and Smooth Signed Distance (SSD) by Taubin [Tau12] arose out of considering smooth priors over the surface. Recent reconstruction methods have become more specialized, in order to target specific shapes, structure, and priors in the missing data or noise. For example, Calakli and Taubin [CT12] does surface reconstruction for point clouds with color information as well as normals.

The work in this thesis is of course based on Taubin [Tau13] which invented the Naïve Non-Convex Hull (NNCH) algorithm. Aside from that, the work of Amenta et al. [ACK01] on the Power Crust presented many interesting theoretical results that can be used to prove NNCH's correctness. Berger and Silva [BS12] comment in passing the usage of a sphere-based Signed Distance Function (SDF) from the MAT, however, their presentation is very short, it is not clear that the idea is the same, they do not seem to have elaborated on it, and the source code is unavailable for comparison.

1.1.2 Medial Representations

Substantial work has been dedicated to extracting medial representations from surfaces, see [SP08] for a thorough overview. There are methods for obtaining medial representations from both meshes and point clouds; only work that applies to the latter will be cited, as the former is considered out of scope for the purpose of this thesis.

Do notice, however, that meshes can be easily sampled into point clouds, so a priori, a method working on a point cloud is strictly more general than one working on a mesh, as the inverse process is inherently more complex and error-prone.

Computation

The computation of these representations is a thoroughly explored topic. Even for surfaces such that $\partial\mathcal{S}$ is C^∞, the MAT can be very unstable under perturbations [CCM97], and this

leads to several numerical instability problems which make accurate estimation a difficult problem. Attali et al. [ABE09] wrote a survey on the topic of its instability alone.

Exact computation of the MAT in general is a hard problem; there exist algorithms of polynomial complexity for structured objects such as polyhedrons [CKM99] and in Constructive Solid Geometry (CSG) [Hof90], but they suffer from numerical instability issues, or have unrealistic assumptions such as fast rational arithmetic.

When working with point clouds, we estimate its Medial Axis Transform (MAT) from the points available. Furthermore, the representation obtained is, of course, finite [1], which means that we have a maximum accuracy given either by the point cloud sampling density or an arbitrary, lower limit, that has been put in place (such as when using Octrees to reduce the density). In practice, the estimation works reasonably well for most common shapes. The notable exception being sharp edges, where it can be seen that the sampling density must be very high in order to obtain a reasonable approximation to $\partial\mathcal{S}$.

There are multiple algorithms for approximating the MAT in the point cloud case, the most popular are derivations of [AB98], where the representations may be accomplished by a proper scaling/filtering of the MAs. Such methods rely on extracting a subset of the Voronoi diagram, which is well defined for densely sampled surfaces, but for point clouds containing missing data the corresponding Voronoi diagram may no longer resemble the MAT.

If approximation approaches are acceptable, more robust methods are available. Ma et al. [MBC12] built an algorithm to approximate the MAT, called Shrinking Ball (SB), as a preprocessing step for Skeletonization; Jalba et al. [JKT13] worked on parallelizing it on the GPU and improving convergence. The most recent results were presented by Peters [Pet18], who adapted the SB to include a denoising heuristic and [Pet14d] [Pet14a] which attempts to approximate the θ Simplified Medial Axis (θ-SMA) [FLM03]. Furthermore, Lam [Lam16] experimented with ways to perform the SB algorithm out-of-core.

Applications

The MAT is a shape representation that holds a very large amount of well-structured, useful information about the surface, which is why it has been extensively used for applications.

The first remarkable application is that of skeletonization, where an input MAT is augmented with connectivity among its MAs' centers (see fig. 1.1), creating what is usually called a Skeleton, or a Structured MAT. Tagliasacchi et al. [Tag+16] wrote an extensive

[1] Moreover, the amount of MAs obtained is usually the same or less as the number of input points

survey of MAT-based skeletonization methods; in particular, Jalba et al. [JKT13] made an algorithm related to our work as it builds upon the SB algorithm that will be introduced in section 3.4.2.

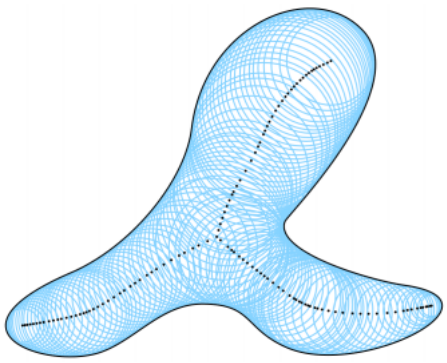


Fig. 1.1: A 2D shape along with its MAT and centers of its MAs, which sketch the skeleton of the shape.

Source: Zhu et al. [Zhu+14]

Decimation or simplification is a task where the intent is to reduce the size of the representation by removing unneeded points for some final task, e.g. reconstruction. Usually, the aim is to preserve the features of the sampled shape as much as possible. This topic has not been very explored in relation to the MAT, which means there is no survey on this particular topic, some pointers for the avid reader are the work by Tam and Heidrich [TH03], Foskey et al. [FLM03], Ma et al. [MBC12], and Peters [Pet14c] [Pet14d].

Other interesting applications, albeit less relevant to this thesis, include curvature motion [Lew+05] [Bor+09] and 3D terrain segmentation [Pet18]. As shown, the MAT is a central concept to many tasks related in one way or another with 3D reconstruction and manipulation in general. It is worth emphasizing that it is also a vastly underdeveloped topic in comparison to many others, and this has to do in part with the complexities surrounding its computation and manipulation, as mentioned earlier.

1.1.3 Signed Distances and Isosurface Extraction

A common pipeline for the generation of 3D meshes is to model an object through the usage of Signed Distance Functions, as will be explained in section 2.2. Such a pipeline can and has been used to perform 3D surface reconstruction from the very start by Hoppe et al. [Hop+92], and has been a common point in many popular techniques such as Poisson Reconstruction [KBH06], MPU [Oht+03], and Moving Least Squares-based techniques like

the work by Alexa et al. [Ale+01] or Guennebaud and Gross [GG07].

The key idea is to model the border $\partial\mathcal{S}$ of the surface $\mathcal{S}$ as the σ level set of a scalar field $f : \mathbb{R}^3 \to \mathbb{R}$, which may be stated explicitly as a function definition[2] or implicitly, for example as a uniform grid of samples[3]. Each of the aforementioned surface reconstruction methods state a set of assumptions on f (concretely, which family of functions can be represented), and develop a method for finding an f within the given family such that its σ level set approximates $\partial\mathcal{S}$ well, informally: $\{\mathbf{x}|f(\mathbf{x}) = \sigma\} \approx \partial\mathcal{S}$.

After f has been found, the surface can be reconstructed by using a method to extract its σ level set and turn it into polygons. Much like in the usual root-finding literature, these methods are usually dependent on the Intermediate Value Theorem for finding the level set, which in turn has translated to certain conditions usually being asked of f (such as continuity), as will be seen later.

Among methods for extraction of 3D surface level sets, the book by Wenger [Wen13] is excellent and has a both extensive and clear exposition of many different methods and principles, going from the classic Marching Cubes by Lorensen and Cline [LC87], to the improved methods that work on the dual representation, like Dual Marching Cubes by Nielson [Nie04] and Dual Contouring by Ju et al. [Ju+02]. There also exist approaches that can generate a mesh directly from the MAT as shown in Delamé et al. [DRF12], however, the methodology is much less understood and significantly more complex. In this thesis, we use MC exclusively, all other methods are mentioned only for completeness.

1.2 Thesis Scope and Outline

The main objective of this thesis is the analysis, implementation, and evaluation of algorithms for 3D surface reconstruction from unstructured point clouds. The chapter 2 serves as the purpose of introducing the reader to basic concepts in the field.

In particular, the focus of this thesis is set on the NNCH algorithm mentioned earlier, and more generally on MAT-based techniques for surface reconstruction. We introduce the MAT and NCH, as well as several of their important theoretical properties in chapter 3. Within that chapter, we also explain the relation between the MAT and the NCH in section 3.3.3, as well as uncover a link to the Power Diagram [ACK01] and Voronoi Diagram in section 3.3.4.

[2] A simple example of this kind of representation is $f(\mathbf{x}) = ||\mathbf{x}||_2 - 1$, where the 0-level set is a sphere

[3] Notice that in this case, the function that determines the grid is not known, only a sample of it is. This technique has been used many methods, such as Poisson Reconstruction [KBH06] and Smooth Signed Distance [Tau12]

The motivation behind choosing the NNCH as an algorithm to work upon, lies in the fact that it is the first method to consider both spheres and planes as atoms when fitting the representation: all other algorithms we are aware of work with either more complex objects fitted locally (such as polynomials, or non-explicit functions), or simple objects fitted globally (like in every MAT-based algorithm).

The NCH indeed has something new to offer, which is that it merges two approaches that have worked well together before: the planes used in multiple reconstruction algorithms such as [Hop+92], and the spheres used in MAT-based reconstruction.

Furthermore, the previous work in similar algorithms show that it can be improved, and moreover that it can be made to work with the large data-sets omnipresent today. This latter assertion is shown by the work of several authors in developing the Shrinking Ball algorithm [MBC12] [JKT13] [Pet18] [Lam16] which we explain in section 3.4; it is upon this work that we built to develop a new fitting algorithm for the NCH, called Shrinking Planes and presented in chapter 4.

We also experimented with several different shapes and different operating conditions to understand the behavior of the SP algorithm, and reached several conclusions on its approximating properties to the NNCH, as shown in section 4.3.

At last, all our conclusions from developing the SP method, as well as several ideas left as future work can be found in chapter 5. Concretely, the contributions of our work can be summarized in:

- Produce a detailed explanation of the NCH, and derive the NNCH algorithm. We also prove many good-behavior properties of the NNCH algorithm.

- Clearly explain the link between the MAT and the NCH, and demonstrate a new link between the NCH, the Power Diagram, and Multiplicatively Weighted Voronoi Diagram.

- Produce a detailed description of how the SB algorithm works, showing some weaknesses of the previous work on it and fixing them in the process.

- Develop a new approximate algorithm for estimating the NCH, based on the SB algorithm. Our method reduces the complexity of estimating the NCH from $\Theta(N^2)$ to expected $\mathcal{O}(N \log N)$.

- Develop an experimental methodology for comparing the NCH approximation quality and evaluate the algorithms proposed against the NNCH, showing that the pro-

posed algorithms achieve extraordinarily good performance at approximating the NCH.

- Run extensive experiments on the reconstruction quality of the NCH, and compare it against Screened Poisson; showing the weaknesses and strengths of NCH-based reconstruction. We also measure and show our performance improvement over the NNCH, and the low memory usage of our algorithm.

- Propose several pathways and ideas for future improvement of NCH-based reconstruction based on our experimentation and survey of the literature on related topics.

2. SURFACE RECONSTRUCTION FROM UNSTRUCTURED POINT CLOUDS

3D Surface Reconstruction usually follows a pipeline of several stages:

1. Obtaining a 3D point cloud by scanning the object. There are many techniques to extract such clouds, from using light with certain patterns to take multiple images together, to special cameras with come embedded with a depth field. Lanman and Taubin [LT09] is a good reference on this topic.

2. Generalizing from the point cloud obtained into a mathematical representation of the surface. This is what is effectively called the reconstruction stage, for which several methods have been mentioned, and it is what the methods in this thesis are for.

3. Polygonalization, concretely, obtaining a polygon mesh out of the mathematical representation chosen for the surface. This is what the isoextraction methods are do, and we will cover more on this later.

4. Visualization, through which we effectively are able to view the output of the previous steps in our computers. For this, we use software such as Meshlab [Cig+08].

The last two steps are optional, and actually dependent on what is the usage for the scanned surface. For example, 3D surface segmentation on LiDAR point clouds is a problem that can be solved without polygonalization.

Concretely, the input to the reconstruction problem is a unstructured point cloud $\mathcal{P}$ sampled from a given surface $\mathcal{S}$ of a three-dimensional object, and the output should be a piecewise-linear approximation of $\mathcal{S}$ (i.e. a polygon mesh).

Definition 2.0.1 (Unstructured Point Cloud)**.** Given $\mathcal{P} = \{\mathbf{p_1}, \ldots, \mathbf{p_N}\}$, and a surface $\mathcal{S}$. $\mathcal{P}$ is called an unstructured point cloud for $\mathcal{S}$ when $\mathcal{P} \subset \partial\mathcal{S}$.

Remark. The point cloud is considered structured when additional information such as connectivity is also provided. In this work, all point clouds considered are unstructured.

As mentioned earlier, $\mathcal{P}$ usually comes from a 3D scanning process, which is a procedure prone to different kinds of errors (see fig. 2.1 for visual examples), following an exposition similar to [Ber+17]:

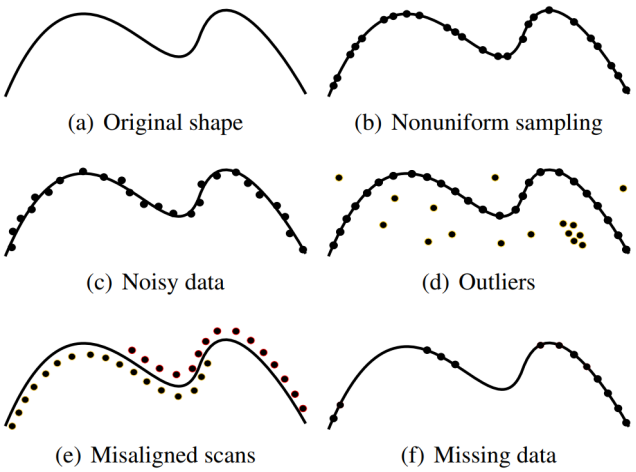
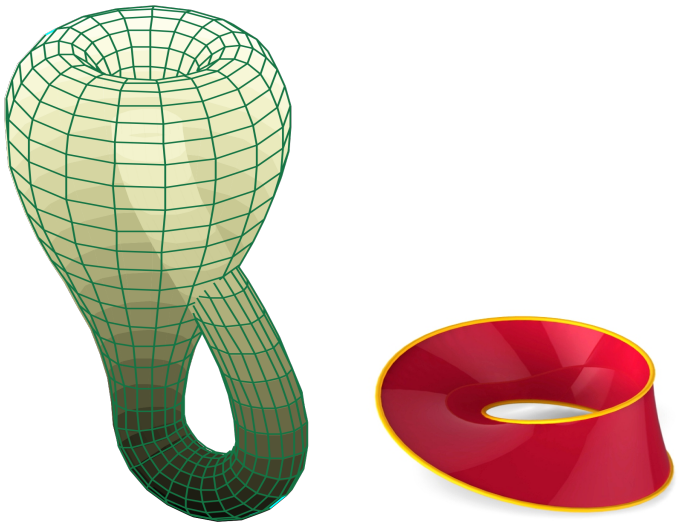


Fig. 2.1: Artifacts present in point clouds, exemplified in 2D.

Source: Berger et al. [Ber+17]

- Non-Uniform Sampling: the points are not uniformly distributed over $\partial\mathcal{S}$. This is usually a byproduct of a bad distance between the object and the scanner, the scanner's orientation, or even shape features.
- Noise: different scanning methods have some amount of measurement noise. It is common for the noise distribution to change in the different parts of the object, as it depends on many scanning artefacts such as the sensor's noise, depth quantization, and the distance or orientation of the surface in relation to the scanner.
- Outliers: unexpected objects characteristics, such as the material's refraction, can lead to severe errors in the point estimates. These are points that are not actually part of $\partial\mathcal{S}$, and ought to be removed or not considered in the reconstruction.
- Misalignment: 3D scanning usually works by taking several scans of the different sides of the object and then merging them together; this process is called Registration. In general, even if camera alignment is optimal, it is almost impossible for the registration process to produce exactly the same point cloud, and hence some amount of misalignment is always to be expected.
- Missing Data: these are due to such factors as limited sensor range, high light absorption, occlusions in the scanning process where large portions of the shape are not sampled, and physical constraints of the scanning device. It is important to distinguish between missing data and non-uniform sampling, as the sampling density is zero in such regions.

These errors are usually handled through regularization: assumptions (priors) are made either on the object being scanned, the input point cloud, or even the scanning method. The variety of priors have led to an equivalent combinatorial explosion in reconstruction methods, each with its own strengths and weaknesses.



(a) A Klein Bottle
Source: Wikipedia

(b) A Möbius Strip
Source: Wikipedia

Fig. 2.2: Example unorientable surfaces

Throughout this thesis, we assume that the surface $\mathcal{S}$ is oriented (i.e. it is possible to make a consistent choice of surface normal vector at every point) and watertight (closed). These are very common assumptions in the interest of reconstruction of surfaces found in every-day life; examples of surfaces that violate the first one are, for example, Klein Bottles and Möbius Strips as seen in fig. 2.2.

Definition 2.0.2 (Unstructured Oriented Point Cloud)**.** Given $\mathcal{P} = \{p_1, \ldots, p_N\}$, $\mathcal{N} = \{n_1, \ldots, n_N\}$, and an oriented surface $\mathcal{S}$. $(\mathcal{P}, \mathcal{N})$ is called an unstructured oriented point cloud for $\mathcal{S}$ when $\mathcal{P}$ is a unstructured point cloud for $\mathcal{S}$, and $\mathcal{N}$ is such that each n_i is the associated unit length orientation vector for p_i, consistently oriented with respect to the surface.

It will also be assumed that we are provided with an unstructured oriented point cloud $(\mathcal{P}, \mathcal{N})$ for reconstruction, similar to fig. 2.3 (from here on we will simply call these point clouds). This is a relatively common requirement as well. Such clouds are produced by laser scanners, structured lighting systems, multiview stereo algorithms, and simulation algorithms.

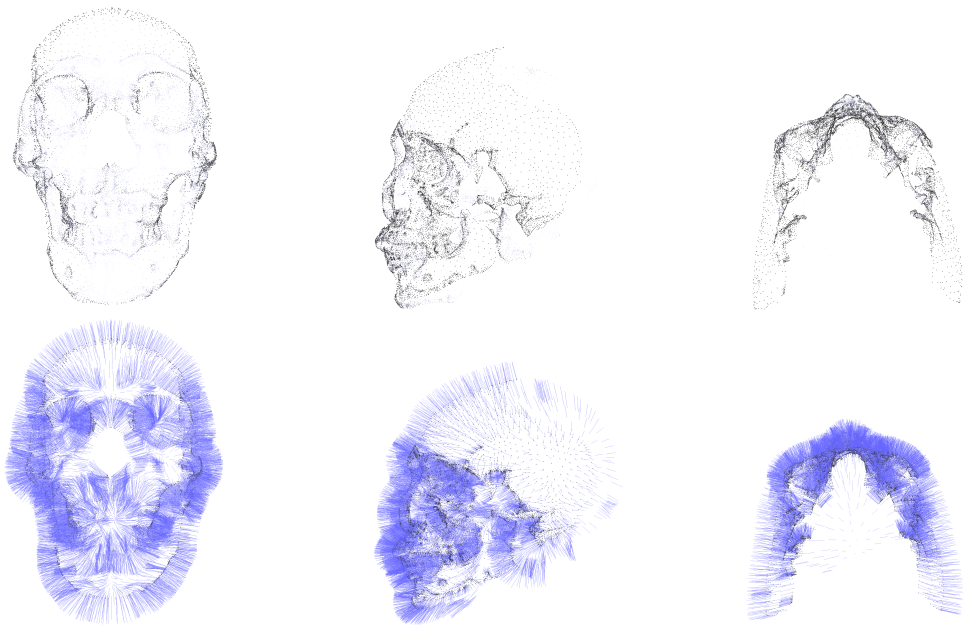


Fig. 2.3: Example of an oriented point cloud of a human skull, from different sides, with and without normal vectors. This cloud suffers from missing data, as well as non-uniform sampling.

2.1 Normal Estimation

When orientation information is unavailable, there are several methods for providing normals for the point cloud, Berger et al. [Ber+17] covers some of them.

Many popular methods are based around fitting a plane to each point by minimizing the mean squared error of the projections onto the plane of nearby points, and then using one of various methods for consistently orienting the normals of the planes. This technique was initially used and popularized by Hoppe et al. [Hop+92], and is solvable in closed form through Principal Component Analysis.

For the purposes of this thesis, these techniques are out of scope; we will always assume that the normals for the point cloud are given to us. When they are not, suffice it to say that this can be achieved using readily available methods from the PCL [RC11] or Meshlab [Cig+08].

2.2 Signed Distance Functions

Definition 2.2.1 (Signed Distance Function)**.** A function $f : \mathbb{R}^k \to \mathbb{R}$ is a Signed Distance Function (SDF) for a surface $\mathcal{S}$ when:

$$f(\mathbf{x}) = \begin{cases} -d(\mathbf{x}, \partial\mathcal{S}) & \mathbf{x} \in \mathcal{S} \\ d(\mathbf{x}, \partial\mathcal{S}) & \mathbf{x} \in \mathcal{S}^c \end{cases}$$

Where $(\mathbb{R}^k, d)$ is a metric space, and we define $d(\mathbf{x}, A) = \inf_{\mathbf{y}\in A} d(\mathbf{x}, \mathbf{y})$ for any set $A \subset \mathbb{R}^k$, particularly in this example, $A = \partial\mathcal{S}$.

Intuitively, an SDF f gives the distance from x to the border of the surface; with it being positive when its in the inside, and negative when its on the outside. Different authors and methods may exchange the signs, but this is the interpretation that will be used throughout this work.

Although the formal definition is as written above, in practice a weaker set of conditions are asked, which depend on what the SDF will be used for, Wenger [Wen13] discusses these conditions in full. For our purposes, an informal characterization is that we need the SDF to be continuously differentiable, separate the inside from the outside through the scalar' sign, and its 0 level set should be the border of the surface we are interested in.

An SDF is a particular representation of a surface, which is usually chosen due to its very convenient manipulation properties; namely:

Lemma 2.2.1. *If f is an SDF for $\mathcal{S}$, then $-f$ is an SDF for $\mathcal{S}^c$.*

Lemma 2.2.2. *If f is an SDF for $\mathcal{S}$, and g is an SDF for $\mathcal{S}'$, then $h(x) = \min(f(x), g(x))$ is an SDF for $\mathcal{S} \cup \mathcal{S}'$.*

Lemma 2.2.3. *If f is an SDF for $\mathcal{S}$, and g is an SDF for $\mathcal{S}'$, then $h(x) = \max(f(x), g(x))$ is an SDF for $\mathcal{S} \cap \mathcal{S}'$.*

These lemmas are the key to what is called Constructive Solid Geometry (CSG). In a nutshell, the most important insight is that it is easy to generate surfaces by combining SDFs, furthermore, we can combine them in a tree structure to generate arbitrarily complex shapes, as shown in fig. 2.4. This representation is also extremely compact and much more flexible in comparison to others: several orders of magnitude less storage are needed for a tree than for a polygonal mesh, and the fact that the abstract representation of the shape is stored, means that it can be modified in the future without requiring complex operations on the polygonal representation.

There are several basic shapes that are well known and can be used as primitives for CSG operations, such as spheres, rectangles, planes, cones, among others. Quilez [Qui11a] published a web page with GLSL implementations of several of these primitives, as well as more complex CSG operations such as twisting and infinite repetition.

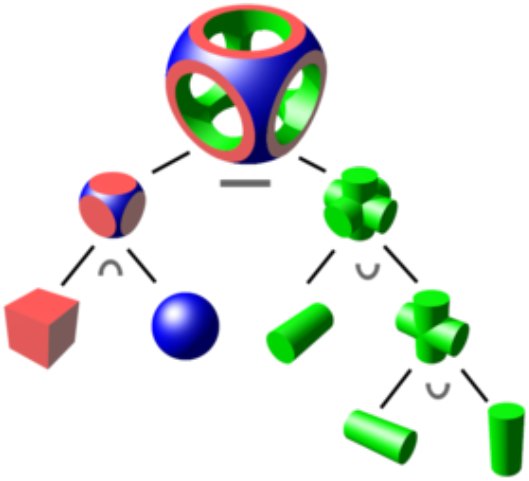


Fig. 2.4: A basic surface generated through boolean CSG operations, shown as a CSG tree.

Source: Wikipedia

2.3 Isosurface Extraction

Once a SDF is given, we will usually be interested in generating a polygonal mesh that represents the border $\partial\mathcal{S}$ of the surface. This process is usually called Isosurface Extraction, Isosurfacing, Contouring, or Isocontouring. Wenger [Wen13] covers several popular techniques in this field in an intuitive yet rigorous manner.

The problem description is as follows: given a SDF $f : \mathbb{R}^3 \to \mathbb{R}$, a value $\sigma \in \mathbb{R}$, we are interested in finding a polygonal mesh $\mathcal{M}$ that approximates the σ-level set of f. For example, $f(x) = 1 - ||x||_2$ is an SDF representing a unit sphere, thus contouring with $\sigma = 0$ should return a polygonal representation of its border; of course, changing the level set that we are using should return spheres that are larger or smaller, according to the change.

The most popular algorithm for isosurface extraction is Marching Cubes (MC) [LC87]. In order to reconstruct the distance field, it requires a uniform grid of samples of f over the interest region; then, the algorithm will pattern-match each cube in the grid and issue one triangle per cube according to the matched pattern. Of course, the grid granularity defines both the number of samples required, and the maximum resolution achievable by the final polygonal mesh.

A 2D example of this idea, called Marching Squares, can be seen in fig. 2.6, assuming $\sigma = 5$. As a first step, the function samples in the scalar grid are classified as $\geq \sigma$ or $< \sigma$ (example in picture (b)); then, the pattern matching table from fig. 2.5 is checked for

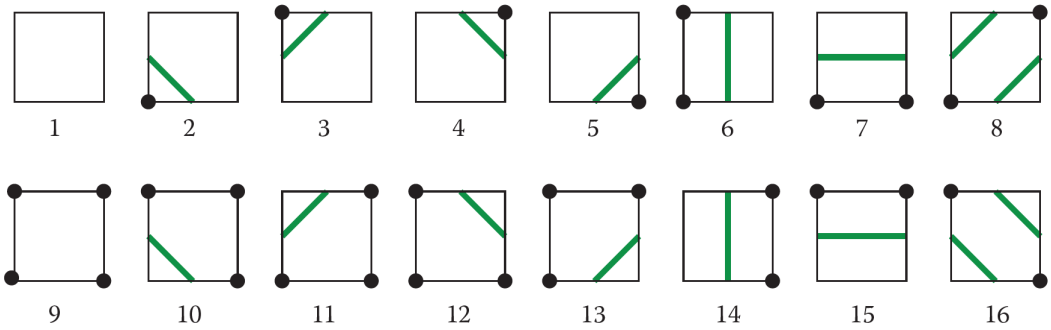


Fig. 2.5: Pattern matching table for Marching Squares. Black dots correspond to vertices at or above the isolevel σ, the other vertices are below the isolevel.

Source: Wenger [Wen13]

each square, generating what is seen in figure (c); at last, linear interpolation is used to approximate where each line should cross the edges, and the final result is shown in (d).

The 3D version has the same steps, but adapted for using cubes instead of squares, and with a much larger pattern matching table that has several complexities[1], see the aforementioned book for more information.

The running time of Marching Cubes is linear in the number of vertices on the scalar grid. Moreover, a 3D uniform grid with K subdivisions has $(K+1)^3$ vertices that need to be sampled. Of course, each sample requires an evaluation of the SDF f, which has an inherent cost. Concretely, although Marching Cubes is a linear time algorithm, generating the uniform grid that it requires can be a very expensive operation, as well as memory intensive if high resolutions are desired.

Marching Cubes has several shortcomings, such as being unable to represent sharp edges, as well as sometimes generating non-manifold meshes or generating cracks in the surface. There have been several modifications and improvements proposed, as well as inherently different techniques proposed. For this thesis, the Marching Cubes implementation in the Point Cloud Library (PCL) [RC11] has been used.

[1] In 3D, there are ambiguous patterns, which are resolved in one way or another as part of the inductive bias of the algorithm, these do not arise in 2D.

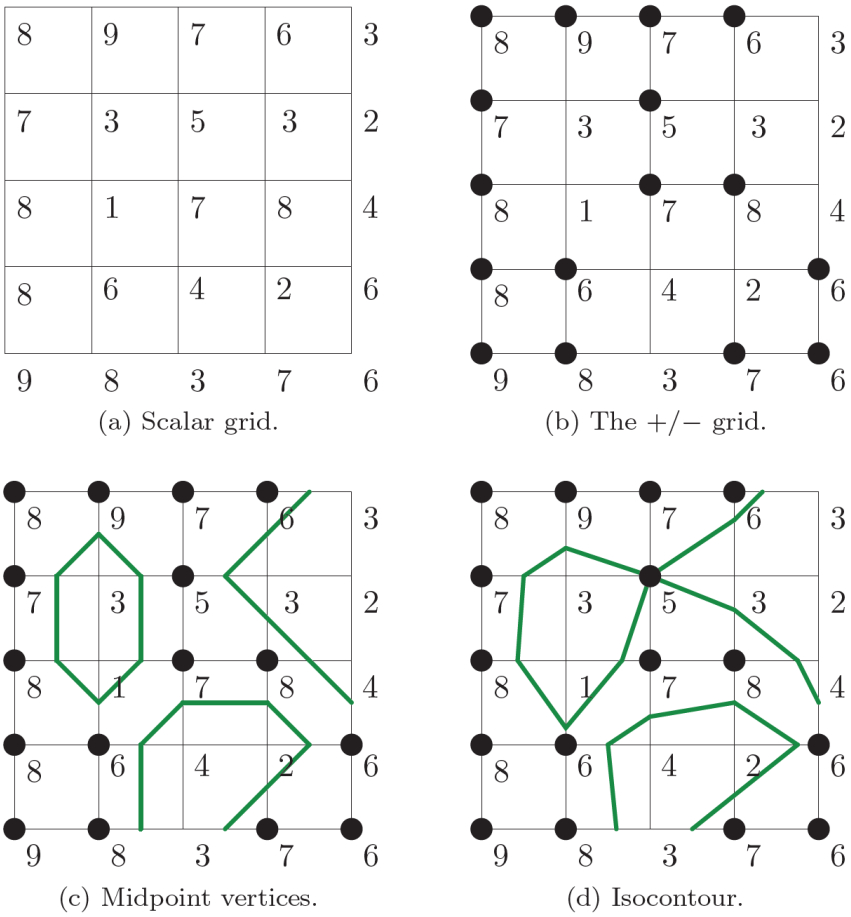


Fig. 2.6: (a) A uniform scalar grid with one sample of $f(\mathbf{x})$ per vertex. (b) In black, vertices that are above or equal to the isolevel. (c) A rough approximation to the curve being fit, as given by the pattern matching table (not shown here), but assuming edge crossings are always at the midpoint. (d) The final isoextraction, after approximating the edge crossings by linear interpolation.

Source: Wenger [Wen13]

3. THE MEDIAL AXIS TRANSFORM AND THE NON-CONVEX HULL

In this chapter, we will formally introduce the representation of a surface as a union of balls, called the Medial Axis Transform (MAT), as well as the representation as a union of balls and planar half-spaces, called the Non-Convex Hull (NCH) (in section 3.1 and section 3.3, respectively).

We will also cover formal definitions that allow us to talk about samples from a surface in a more precise manner in section 3.2; these will be useful when discussing several concepts in the field.

Aside from that, in section 3.3.3 we will explain the relationship between the NCH and the MAT, as well as why the NCH has properties that make it a better fit for the reconstruction problem. Furthermore, we will explore the relationship between the NCH, the Power Diagram, and the Voronoi Diagram in section 3.3.4.

At last, we will introduce many practical algorithms for computing the MAT and the NCH in section 3.4. In particular, the Naïve Non-Convex Hull (NNCH) algorithm will be introduced in section 3.4.1, the Shrinking Ball (SB) algorithm for computing the MAT in section 3.4.2, and the noise resistant variant of the SB algorithm, the Denoising Shrinking Ball (DSB) algorithm, in section 3.4.3.

3.1 Medial Axis Transform

There are several different ways to define the MAT. The exposition that follows is inspired on multiple sources, suffice it to say that we are going to expose just one of many equivalent characterizations; for a complete overview, the book by Siddiqi and Pizer [SP08] can be a valuable resource. To make definitions easier, we take the following set of inscribed disks, as in [CCM97]:

$$\mathcal{D}(\Omega) = \{B_r(\mathbf{p}) | \mathbf{p} \in \Omega, r \in \mathbb{R}_{\geq 0}, B_r(\mathbf{p}) \subset \Omega\}, \Omega \subset \mathbb{R}^k$$

Remark. $B_r(\mathbf{p})$ is considered to be the closed ball of radius r, centered at p. If $r = 0$, then $B_r(\mathbf{p}) = \{p\}$.

Notice that $\mathcal{D}(\Omega)$ is the set of all balls contained inside Ω, where Ω is any connected and bounded subset of $\mathbb{R}^k$; this set is naturally uncountable whenever Ω is not finite. For

example, if $k = 1$ and we take $\Omega = [0, 1]$, $\mathcal{D}(\Omega)$ is the set of all intervals contained within $[0, 1]$. When going into higher dimensions, such as $k = 2$ or $k = 3$, this becomes a set of balls and spheres, respectively. Of course, $\cup_{A \in \mathcal{D}(\Omega)} A = \Omega$ by definition, so the set indeed covers all of Ω.

Definition 3.1.1 (Core)**.** $\mathbf{CORE}(\Omega)$ is the set of maximal inscribed disks in Ω, each one called Medial Atom (MA) and defined as follows:

$$\mathbf{CORE}(\Omega) = \{B_r(\mathbf{p}) \in \mathcal{D}(\Omega) | (B_s(\mathbf{q}) \in \mathcal{D}(\Omega) \wedge B_r(\mathbf{p}) \subset B_s(\mathbf{p})) \implies B_r(\mathbf{p}) = B_s(\mathbf{q})\}$$

The $\mathbf{CORE}(\Omega)$ is a filter where we remove all non-maximal subsets. In the previous example of $[0, 1]$, what happens is that $\mathbf{CORE}(\Omega) = \{B_{0.5}(\mathbf{0.5})\}$; since $B_{0.5}(\mathbf{0.5}) = [0, 1]$, indeed this is the maximal representation. This example translates equivalently into $k = 2$ and $k = 3$ by considering a unit ball or sphere, respectively.

Definition 3.1.2 (Medial Axis)**.** $\mathbf{MA}(\Omega)$ is the set of centers of the Medial Atoms in $\mathbf{CORE}(\Omega)$:

$$\mathbf{MA}(\Omega) = \mathrm{cl}(\{\mathbf{p} \in \Omega | B_r(\mathbf{p}) \in \mathbf{CORE}(\Omega)\})$$

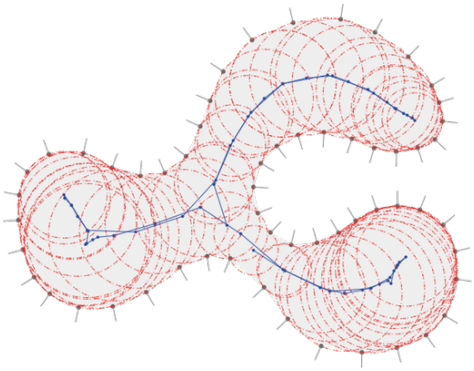
Notice here that we taking the closure *cl* of the ball's centers, concretely including the limit points. This trivially makes the MAT closed, and even compact if Ω is bounded [SPW96], which are facts that will be useful later on when considering what happens with sharp edges.

Definition 3.1.3 (Medial Axis Transform)**.** $\mathbf{MAT}(\Omega)$ is the set of pairs consisting of the center and radius of the disks in $\mathbf{MA}(\Omega)$.

Remark. By definition, two Medial Atoms can't have the same center (this follows from definition 3.1.1). Hence, the mapping $\mathbf{MA}(\Omega) \to \mathbf{MAT}(\Omega)$ giving the radius for each center is well defined, one-to-one, and onto. This makes clear the notion that they are just different representations of the same object.

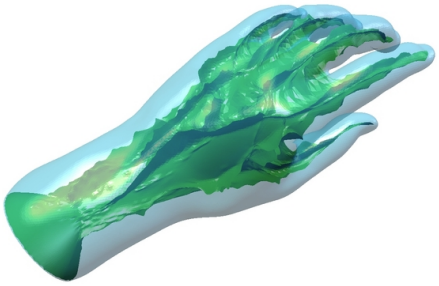
As an example, picture (a) in fig. 3.1 shows a surface along with its sampled point cloud and normals. Inside of it, its Medial Atoms are shown, and the center of each ball is shown in blue. Furthermore, they are connected to show the MAT's interpretation as a skeleton of the shape. Picture (b) shows an example in 3 dimensions; in this case, the MAT is not shown explicitly, but the green material inside the object are precisely the sheets that are composed by connecting the MAT's centers.

From here on, the literature splits in two, one part studies the 2D MAT, while the other the 3D MAT. The reason for this divergence is that they are inherently different



(a) A 2D figure with its sampled point cloud and outward pointing normals, the inscribed balls from its MAT, and the centers connected by lines after a Skeletonization process.

Source: Peters [Pet18]



(b) A 3D model for a hand, and its connected MAT centers.

Source: Shin Yoshizawa

Fig. 3.1

objects; the reader with an interest in the details may want to look at [SPW96; CCM97], where further pointers may be found. The differences summarize in that the 2D MAT is much better understood than the 3D MAT, significantly easier to approximate, and has multiple properties that help its algorithmic manipulation.

It is also usual in the literature to divide the MAT into Exterior (or Outer) and Interior (or Inner), where the Interior MAT refers to the definitions given earlier, while the Exterior is defined equivalently for Ω^c. A intuitive analogy is that the Interior MAT corresponds to building a shape out of balls, while the Exterior MAT corresponds to carving out balls in a block of marble: in both cases we are generating the same shape (when the surface is infinitely sampled), but one method is "additive" and the other is "subtractive".

For what follows, $\Omega \subset \mathbb{R}^3$; although much of this exposition applies to $\mathbb{R}^k$, if not any metric space. We are interested in a few properties of the 3D MAT that will simply be used throughout the thesis:

- Completeness, Recomposition, or Invertibility. The union of Ω's MAs is exactly Ω, allowing a full reconstruction directly from the atoms [Blu67].
- Uniqueness. There is a unique MAT for a given Ω, and furthermore it is invariant with respect to the coordinate system [She95].
- Topological Equivalence. The MAT is topologically (homotopically) equivalent to

the object. That is, all the features of the given object such as holes, slots, etc., are retained in its MAT [SP08].

- Hierarchical Decomposition. The dimensionality of the MAT is lower than that of Ω. The 2D MAT transforms planar figures into lines, and 3D MAT decomposes an object into surface patches [SP08]. Furthermore, these patches enable a hierarchical traversal of the different parts of the surface [Blu67].

- Symmetry. The MAT is centered in the middle of the object, by construction [Blu67].

- Instability. Small changes in the boundary of an object may cause a large change in its MAT. Sometimes a large number of variations on the object boundary results in a discontinuous MAT [ABE09; CCM97].

- Continuity or Smoothness. The discussion and results around its continuity are varied and dependant on several conditions; however, each sheet in its hierarchical decomposition is known to be at least $\mathcal{C}^2$ [SP08; SPW96; CCM97].

Technicalities lie behind all the aforementioned properties, but those will not be discussed here for brevity. Again examining fig. 3.1, it is easy to see almost all properties mentioned: it is clear that the union of the balls generates the shape, that it is topologically equivalent, and the smoothness and symmetry are self-evident from looking at the blue skeleton connecting the centers of the MAs.

As to the hierarchical decomposition, in this particular case it refers to the three lines that separate from the middle point; those effectively "separate" the shape into three skeleton parts.

Results to exemplify the instability of the MAT are very involved constructions, and can be seen in Attali et al. [ABE09]; suffice it to say that it is possible to construct objects with infinitely many skeleton branches just by adding slight sinusoid-like deformations to the border. In practice, what happens is that a skeleton branch could potentially be added by just a little bit of noise; this indeed is something that can happen in any 3D scan.

3.2 Sampling

There are several conventional restrictions when talking about point clouds, usually having to do with the maximum error allowed, or the sampling density, and so on and so forth. Here, we give some definitions that are commonly used.

Definition 3.2.1 (δ-noisy Sample)**.** Let $\mathcal{P} = \{\mathbf{p_1}, \ldots, \mathbf{p_N}\}$ be a point cloud for $\mathcal{S}$. It is assumed that each $\mathbf{p_i} = \hat{\mathbf{p}}_\mathbf{i} + \mathbf{e_i}$, where $\hat{\mathbf{p}}_\mathbf{i} \in \partial\mathcal{S}$, and $\mathbf{e_i}$ represents the sampling error. $\mathcal{P}$ is called a δ-noisy sample of $\mathcal{S}$ when:

$$\forall i \in [N] : ||\mathbf{e_i}|| < \delta$$

In other words, the maximum error achieved when measuring the surface must be less than δ.

Remark. We take $[N] = \{1, \ldots, N\}$ as a shorter, clearer notation.

Definition 3.2.2 (ρ-dense Sample)**.** Let $\mathcal{P}$ be a point cloud for $\mathcal{S}$. $\mathcal{P}$ is said to be ρ-dense when:

$$\forall \mathbf{x} \in \partial\mathcal{S} : B_\rho(\mathbf{x}) \cap \mathcal{P} \neq \phi$$

In other words, when each point in the surface's border has at least one sample closer than ρ.

These two definitions can be hard to understand when taken together, as it is not clear how to combine the noise with the sampling density. In general, if $\mathcal{P}$ is δ-noisy, we will call it ρ-dense only when it is so after removing the noise.

Although these definitions are very intuitive to understand and work with, the ρ-dense requirement is not very good, as it does not account for the differences in the level of detail of each part of the surface. The following definitions, due to Amenta and Kolluri [AK00], are much better suited and used throughout the literature for its nice properties:

Definition 3.2.3 (Local Feature Size (LFS))**.** Given a point $\mathbf{w} \in \mathcal{S}$,

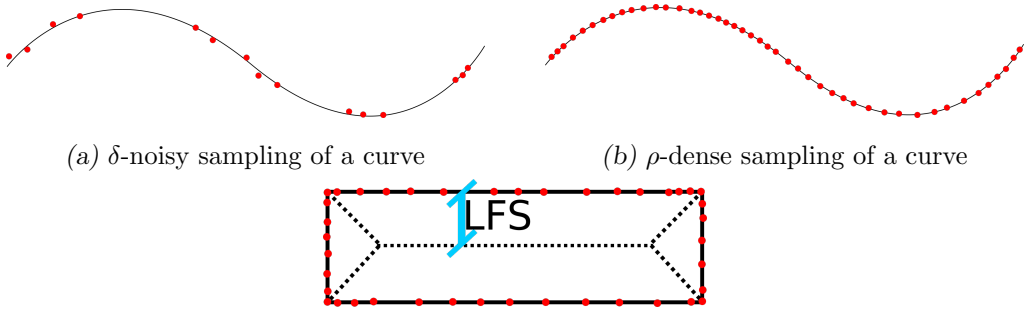
$$\mathbf{LFS}(\mathbf{w}) = d(\mathbf{w}, \mathbf{MA}(\mathcal{S})) = \inf_{\mathbf{c} \in \mathbf{MA}(\mathcal{S})} \{d(\mathbf{w}, \mathbf{c})\}$$

. In other words, it is the distance from the point to the nearest point of the medial axis of $\mathcal{S}$.

Definition 3.2.4 (ϵ-sample)**.** A point cloud $\mathcal{P}$ is a ϵ-sample of $\mathcal{S}$ if for any $\mathbf{x} \in \partial\mathcal{S}$:

$$d(\mathbf{x}, \mathcal{P}) = \mathcal{O}(\epsilon)\mathbf{LFS}(\mathbf{x})$$

Where $\mathcal{O}(\epsilon)$ is taken as the standard Computer Science usage of the collectively called Bachmann–Landau notation.



(a) δ-noisy sampling of a curve

(b) ρ-dense sampling of a curve

(c) ϵ-sample of a rectangle, with $\epsilon \approx 1$. The dotted lines correspond to where the MAT's centers should be. The cyan line shows what is meant by the distance to the MAT from the border.

Fig. 3.2: Examples of δ-noisy, ρ-dense, and ϵ-sample, in order to emphasize the difference in the conditions.

Concretely, the Local Feature Size (LFS) tells us how far we are from a surface's skeleton. For example, if we imagine a pipe, its skeleton runs down the middle as a line; thus, the local feature size is going to be the radius of the pipe.

The ϵ-sample embodies the idea that a sample should be more dense in areas of the surface with finer details. Notice that this definition is very different than ρ-dense, as an ϵ-sample can be less dense than ρ in certain parts of the surface, and require more samples in others.

The definition is very convenient as well because it is often used to prove that a MAT approximation algorithm converges to it as $\epsilon \to 0$, and can also be used as a point decimation heuristic to ensure a minimal representation [MBC12] [JKT13] [Pet18]. The LFS also has several useful properties, such as being Lipschitz continuous, which are developed throughout several papers, such as Amenta et al. [ACK01], [AB98] and [AK00].

In fig. 3.2, we show an example of each sampling condition and what it is meant by each one of them. In picture (a), corresponding to a δ-noisy sampling, notice how points are not evenly distributed, but are always within a given distance, δ, from their projection upon the curve. On the other hand, if we look at picture (b), a ρ-dense sample, the surface has got to have a sample sufficiently near; this implies a given amount of points are needed, and forces sampling. The last picture, (c), which corresponds to an ϵ-sample, we can see how the sampling density is greater at the corners than in the rest of the figure, and furthermore how there is a very loose requirement for a uniformity.

3.3 Non Convex Hull

3.3.1 Core Definitions

The following exposition roughly follows Taubin [Tau13], albeit with more examples to aid understanding, and slight differences that lead to the same definitions.

Definition 3.3.1 (Half-Space)**.** Given $X \subset \mathbb{R}^k$ and $f : X \to \mathbb{R}$, $H_f = \{x \in X | f(x) \leq 0\}$ is a half-space

Definition 3.3.2 (Supporting Half-Space)**.** Given a point cloud $\mathcal{P} = \{\mathbf{p_1}, \ldots, \mathbf{p_N}\}$, H_f is a supporting half-space for $\mathcal{P}$ when:

- $\mathcal{P} \subseteq H_f$
- $\exists \mathbf{x} \in \mathcal{P} : f(\mathbf{x}) = 0$

Concretely, all of $\mathcal{P}$ is inside the half-space, and at least one point is in its border.

Definition 3.3.3 (Linear Half-Space)**.** Given a half-space H_f, it is called linear if f is a linear function.

The Convex Hull (CH) of the set $\mathcal{P}$, $\mathbf{CH}(\mathcal{P})$, can be defined as the intersection of all the supporting linear half spaces for $\mathcal{P}$. Since a linear half space is a convex set, and convexity is preserved by intersection, $\mathbf{CH}(\mathcal{P})$ is also convex.

Definition 3.3.4 (Oriented Convex Hull)**.** Let $(\mathcal{P}, \mathcal{N})$ be an oriented point cloud. Define $f_i(\mathbf{x}) = \langle \mathbf{n_i}, \mathbf{x} - \mathbf{p_i} \rangle$. Then:

$$\mathbf{OCH}(\mathcal{P}, \mathcal{N}) = \bigcap_i H_{f_i}$$

Remark. $f_i(x)$ as defined in the Oriented Convex Hull (OCH) is positive when x is on the inward side of the half-space, as defined by plane centered at $\mathbf{p_i}$ with normal $\mathbf{n_i}$. Under this definition H_{f_i} is in fact the other side, including the border.

Remark. The oriented convex hull is literally fitting planes pointing inwards and computing the intersection of their complements

Remark. If the orientation of the points is reversed, the result is completely different; furthermore, the intersection can be empty. For example, taking $\mathcal{P} = \mathcal{N} = \{(-1, 0); (1, 0)\}$, then $\mathbf{OCH}(\mathcal{P}, -\mathcal{N}) = \{(x, y) \in \mathbb{R}^2 | -1 \leq x \leq 1\}$, while $\mathbf{OCH}(\mathcal{P}, \mathcal{N}) = \phi$.

Remark. The OCH is a convex set, as it is an intersection of linear half-spaces, which are themselves convex sets.

Definition 3.3.5 (Non-Convex Hull)**.** Let $(\mathcal{P},\mathcal{N})$ be an oriented point cloud, and $\rho = \{\rho_1,\dots,\rho_N\}$ such that $\rho_i \geq 0$. Define $f_i^{\rho_i}(\mathbf{x}) = \langle \mathbf{n_i}, \mathbf{x}-\mathbf{p_i}\rangle - \rho_i||\mathbf{x}-\mathbf{p_i}||^2$. Then:

$$\mathbf{NCH}(\mathcal{P},\mathcal{N},\rho) = \bigcap_i H_{f_i^{\rho_i}}$$

Notice that all the half-spaces involved are supporting by construction. Furthermore, define the set of Non-Convex Hull Atom (NCHA)s:

$$\mathbf{NCHA}(\mathcal{P},\mathcal{N},\rho) = \{(\mathbf{p_i},\mathbf{n_i},\rho_i)|1 \leq i \leq N\}$$

Remark. When $\rho_i = 0$, $f_i^0(\mathbf{x})$ is equivalent to the f_i used in the OCH.

‘

Lemma 3.3.1. *When $\rho_i > 0$, $f_i^{\rho_i}(\mathbf{x})$ is a SDF for the compliment of a sphere centered on $\mathbf{p_i} + r_i\mathbf{n_i}$ with radius $r_i = \frac{1}{2\rho_i}$. Concretely:*

$$f_i^{\rho_i}(\mathbf{x}) = \frac{1}{2r_i}\{r_i^2 - ||\mathbf{x}-(\mathbf{p_i}+r_i\mathbf{n_i})||^2\}$$

Proof. First, notice that:

$$\begin{aligned}||\mathbf{x}-(\mathbf{p}+r_i\mathbf{n})||^2 &= \langle \mathbf{x}-\mathbf{p}-r_i\mathbf{n}, \mathbf{x}-\mathbf{p}-r_i\mathbf{n}\rangle \\ &= \langle \mathbf{x}-\mathbf{p}, \mathbf{x}-\mathbf{p}\rangle + \langle \mathbf{x}-\mathbf{p}, -r_i\mathbf{n}\rangle + \langle -r_i\mathbf{n}, \mathbf{x}-\mathbf{p}\rangle + \langle -r_i\mathbf{n}, -r_i\mathbf{n}\rangle \\ &= ||\mathbf{x}-\mathbf{p}||^2 + (-2r_i)\langle \mathbf{n}, \mathbf{x}-\mathbf{p}\rangle + r_i^2\end{aligned}$$

Replacing in the formula:

$$\begin{aligned}f_i^{\rho}(\mathbf{x}) &= \frac{1}{2r_i}(r_i^2 - (||\mathbf{x}-\mathbf{p_i}||^2 + (-2r_i)\langle \mathbf{n_i}, \mathbf{x}-\mathbf{p_i}\rangle + r_i^2)) \\ &= \frac{1}{2r_i}(-||\mathbf{x}-\mathbf{p_i}||^2 + (2r_i)\langle \mathbf{n_i}, \mathbf{x}-\mathbf{p_i}\rangle) \\ &= \langle \mathbf{n_i}, \mathbf{x}-\mathbf{p_i}\rangle - \frac{1}{2r_i}||\mathbf{x}-\mathbf{p_i}||^2\end{aligned}$$

Which is exactly the original definition. The important detail here is that the SDF for the sphere is undefined when $r_i = 0$, which is why the other representation is taken.

To corroborate that this is indeed the complement of a sphere, notice that:

$$\begin{aligned} f_i^{\rho}(\mathbf{x}) &\geq 0 \\ r_i^2 &\geq ||\mathbf{x} - (\mathbf{p_i} + r_i\mathbf{n_i})||^2 \\ r_i &\geq ||\mathbf{x} - (\mathbf{p_i} + r_i\mathbf{n_i})|| \end{aligned}$$

Which is the definition of being inside the mentioned sphere. Via the same methodology, we can verify that the 0 level set of $f_i^{\rho_i}$ corresponds to the border of the sphere. □

Since we have established the relation between planes, spheres, radii, and ρ, we will refer to each of them equivalently. We will denote $f_i^{r_i}(\mathbf{x})$ (where r_i is a radius, possibly infinite) as equivalent to $f_i^{\rho_i}(x)$ with $\rho_i = 0$ when $r_i = \infty$, and $\rho_i = \frac{1}{2r_i}$ when it is positive. When all variables are clearly fixed, $f_i(\mathbf{x})$ will be used to denote $f_i^{r_i}(\mathbf{x})$.

Remark. It is important to notice that similarly to the OCH, the intersection happens between the complements of the shapes represented, as H is taking only negative values, and our functions are positive outside the surface.

Definition 3.3.6 (NCH SDF)**.** $f(\mathbf{x}) = \max_{1\leq i\leq N} f_i^{r_i}(\mathbf{x})$ is also an SDF, representing the intersection of all $f_i^{r_i}$ (intersection of complements). Notice that each one of those objects may be either a sphere, or a hyperplane.

Lemma 3.3.2 (Non Convex Hull and SDF equivalence)**.** *Let $(\mathcal{P}, \mathcal{N})$ be an oriented point cloud, ρ defined for each point, f the corresponding NCH SDF, then: $H_f = \mathbf{NCH}(\mathcal{P}, \mathcal{N}, \rho)$. Furthermore, the zero-level set of f coincides with the boundary of the NCH.*

Proof. Notice that $\mathbf{x} \in H_f \iff f(\mathbf{x}) \leq 0 \iff f_i(\mathbf{x}) \leq 0 \forall i \in [N] \iff \mathbf{x} \in \bigcap_i H_{f_i} \iff \mathbf{x} \in \mathbf{NCH}(\mathcal{P}, \mathcal{N}, \rho)$.

The fact that the boundary matches follows from the fact that $f(\mathbf{x})$ is constructed as the intersection and the SDF preserves the border. □

Corollary 3.3.1. *The gradient of the basis functions is*

$$\nabla f_j^{r_j}(\mathbf{x}) = \mathbf{n_j} - 2\rho_j(\mathbf{x} - \mathbf{p_j})$$

And more importantly, it coincides with the normal at the point:

$$\nabla f_j^{r_j}(\mathbf{p_j}) = \mathbf{n_j}$$

Moreover, $||\nabla f_j^{r_j}(\mathbf{x})|| = 1$ whenever $f_j^{r_j}(\mathbf{x}) = 0$.

Definition 3.3.7. We define $J(\mathbf{x}) = \{j : f_j^{r_j}(\mathbf{x}) = f(\mathbf{x})\}$, that is, the set of point indices whose basis functions achieve the maximum for a given point.

Remark. Given any $\mathbf{x}$, $J(\mathbf{x}) \neq \phi$, since every point has at least one function that sets the maximum.

Lemma 3.3.3. *f is differentiable almost-everywhere. When $J(\mathbf{x}) = \{j\}$, it is differentiable, and its gradient is uniquely defined as*

$$\nabla f(\mathbf{x}) = \nabla f_j^{r_j}(\mathbf{x})$$

Proof. The first thing to notice is that all $f_j^{r_j}(\mathbf{x})$ are $\mathcal{C}^\infty$, i.e. smooth, which implies that they are also locally Lipschitz. Now we can use Lemma 2.1 from [Hei14] to establish that $f(\mathbf{x})$ is locally Lipschitz, as it is a supremum over a finite set of functions. Finally, Rademacher's theorem, proven in the same notes, tells us that $f(\mathbf{x})$ is almost-everywhere differentiable.

Now we need to figure out what the derivative of $f(\mathbf{x})$ actually is. In principle, $J(\mathbf{x}) \neq \phi$ for sure, as every point has at least one function that determines the maximum. It may be a singleton, or it may be of size up to N (the number of points).

In the scenario in which $J(\mathbf{x}) = \{j\}$, the directional derivative of $f(\mathbf{x})$ is exactly the directional derivative of $f_j^{r_j}(\mathbf{x})$, which implies that $\nabla f(\mathbf{x}) = \nabla f_j^{r_j}(\mathbf{x})$.

Such is the case for most points, except for the boundaries where the value of $f(\mathbf{x})$ is defined by a different function. The case when there are many points that achieve the maximum is precisely the reason why we prove almost-differentiability. □

Remember from definition 3.3.5 that all the basis functions $f_j(\mathbf{x})$ are supporting for $\mathbf{p_j}$ (i.e. $f_j(\mathbf{p_j}) = 0$), regardless of the choice of ρ_j. Furthermore, the statements in corollary 3.3.1 along with lemma 3.3.3 prove that $\nabla f_j(\mathbf{p_j}) = \mathbf{n_j}$. Thus, if we look at the surface defined by the zero-level set of $f(\mathbf{x})$, it is both interpolating and matches at the normals for all points in the cloud; and this is independent of the particular ρ that we choose.

Nevertheless, nothing has been said up to this point on what $\rho = \{\rho_1, \ldots, \rho_N\}$ should look like. The way Taubin [Tau13] defines it is basically so that we can always ensure that, for a given oriented point cloud $(\mathcal{P}, \mathcal{N})$ and point $\mathbf{p_i}$:

1. The point is at the border of the SDF, i.e. $f_i^{\rho_i}(\mathbf{p_i}) = 0$.

2. The normal of the SDF at the point coincides, i.e. $\nabla f_i^{\rho_i}(\mathbf{p_i}) = \mathbf{n_i}$.

3. Other points are left outside of the sphere, or at its border, i.e. $f_i^{\rho_i}(\mathbf{p_j}) \leq 0$ for all $j \neq i$.

4. r_i is maximal and $\rho_i \geq 0$.

The reasoning behind this choice is simple: imagine a watertight surface $\mathcal{S}$ with a single connected component and its sample point cloud. If we take any given point $\mathbf{p}$, then there is at least one Medial Atom that is tangent to it, explaining the first requirement; by definition, the atom will be maximal, explaining the fourth; and inscribed in the surface, which accounts for the third. The only missing explanation is for the second one, and this is just to guarantee that the ball is inscribed inside the object, as the center of the ball will be in the same direction as the normal.

The first and second requirements are satisfied by the choice of $f_i^{\rho_i}$ itself, so there is no need to do anything. The interesting piece is how to choose ρ_i. However, this is a straightforward derivation if we ask that the third property be true:

$$\begin{aligned} f_i^{\rho_i}(\mathbf{p_j}) &\leq 0 \\ \langle \mathbf{n_i}, \mathbf{p_j} - \mathbf{p_i} \rangle - \rho_i ||\mathbf{p_j} - \mathbf{p_i}||^2 &\leq 0 \\ \rho_{ij} = \frac{\langle \mathbf{n_i}, \mathbf{p_j} - \mathbf{p_i} \rangle}{||\mathbf{p_j} - \mathbf{p_i}||^2} &\leq \rho_i \end{aligned}$$

Thus, we have that $\rho_i \geq \max_{1 \leq j \leq N, j \neq i} \rho_{ij}$. From this derivation, it would seem valid to set ρ_i to the maximum of the ρ_{ij}s. However, there is a border case: if $\langle \mathbf{n_i}, \mathbf{p_j} - \mathbf{p_i} \rangle < 0$ for all $j \neq i$ (i.e. there aren't any sampled points in the half-space centered at $\mathbf{p_i}$ and pointing inwards), then $\rho_i < 0$, which is an invalid configuration as per the last properties.

This reason is precisely why 0 is a valid value for ρ_i. In these cases, we just set $\rho_i = 0$, which means that we are fitting a hyperplane. Thus, the definition of ρ_i is:

$$\rho_i = \max\{0, \max_{1 \leq j \leq N, i \neq j} \rho_{ij}\}$$

Under these definitions, fig. 3.3 shows an example NCH fit for a 2D shape. It is important to notice the limitations of the OCH in (C), and how those are overcome through the use of circles in (F).

3.3.2 Symmetric Non-Convex Hull

Thus far, we have worked with an oriented point cloud $(\mathcal{P}, \mathcal{N})$ with a given normal orientation, we will call this NCH SDF as $f^+(\mathbf{x})$. From here on, we will call $f^-(\mathbf{x})$ to the NCH

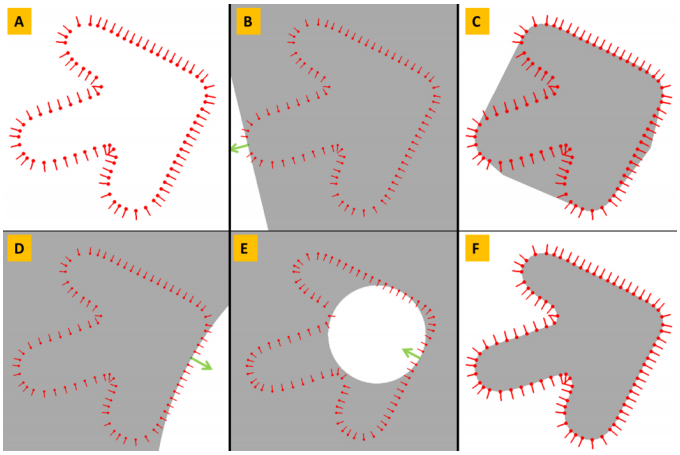


Fig. 3.3: A: A 2D oriented point cloud. B: A supporting linear half space for one of the oriented points. C: The OCH of the point cloud. D: An outside supporting circle for one of the points. E: Inside supporting circles are obtained by inverting the orientation vectors. F: The NCH of the oriented point cloud is the intersection of the complement of all the outside supporting circles.

Source: Taubin [Tau13]

SDF as defined in the same way, but with the normals pointing in the opposite direction. The relationship between these two functions is similar to what happens with the Inner and Outer MAT.

Definition 3.3.8 (Symmetric NCH)**.** We call

$$\hat{f}(x) = \frac{f^+(x) - f^-(x)}{2}$$

The Symmetric NCH SDF.

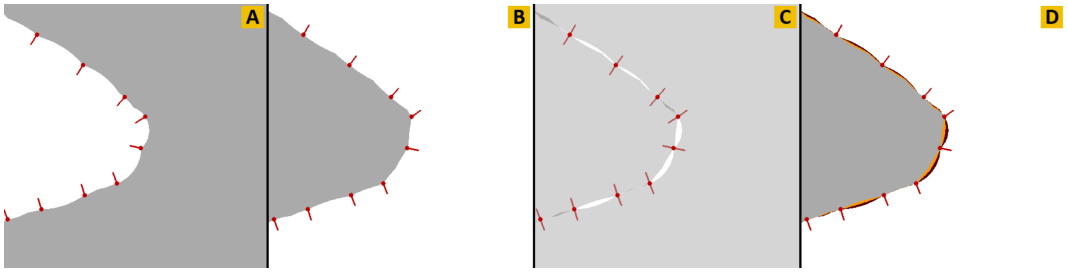


Fig. 3.4: A: Inner NCH. B: Outer NCH. C: Intersection area in the middle. D: The new border determined by the Symmetric NCH, and how it separates the two original borders.

Source: NCH Presentation Slides at SIGGRAPH Asia 2013

Remark. The Symmetric version requires computing two different maximums instead of one. Although from a time complexity point of view this does not make any difference, in practice evaluation is often much more expensive, even though some computation can be reused.

Remark. This is not equivalent to any particular CSG operation between the surfaces determined by the SDFs. Nevertheless, it determines a zero level set that is very similar to the intersection of the NCH surfaces. Furthermore, the definition is consistent with the original NCH definition. See fig. 3.4 for a clear example.

Remark. This formulation is not equivalent to $g(\mathbf{x}) = \max_{1 \leq i \leq N}(\frac{f_i^+(\mathbf{x}) - f_i^-(\mathbf{x})}{2})$. They lead to different reconstructions and, although this variant works for some objects, it eventually fails in subtle ways.

It can be proven that each basis function defined as $\frac{f_i^+(\mathbf{x}) - f_i^-(\mathbf{x})}{2}$ has the same zero level set a basis function for the same point and normal, but with $\rho_i = \frac{\rho_i^+ - \rho_i^-}{2}$. Concretely, such an operation is basically an adjustment on the Inner radius.

The above statement implies that properties 3 and 4 from the previous section do not hold when using $g(\mathbf{x})$, which explains why reconstruction fails in this particular case.

3.3.3 Relationship with the MAT

As mentioned earlier, the key difference between the MAT and the NCH is that the latter is also able to use planes as "Medial Atoms". If we take away this freedom from the NCH, what we are left is a shape represented as a union of balls, which is exactly what the MAT is. Taubin [Tau13] first established this relation, and the following lemma is a small proof of the discussion in the original paper:

Lemma 3.3.4. *Let $(\mathcal{P}, \mathcal{N})$ be an oriented point cloud, with ρ suitably defined such that $\rho_i > 0$ for all i (i.e. no plane fits are allowed). Also let $\mathcal{S}^c = \mathbf{NCH}(\mathcal{P}, \mathcal{N}, \rho)$. Then, there is a surjective function $g : \mathbf{NCHA}(\mathcal{P}, \mathcal{N}, \rho) \to \mathbf{MAT}(\mathcal{S})$.*

Proof. We define $g(\mathbf{p_i}, \mathbf{n_i}, \rho_i) = (\mathbf{p_i} + r_i \mathbf{n_i}, r_i)$. Take any $B_r(\mathbf{p}) \in \mathbf{MAT}(\mathcal{S})$; by construction, $\exists \mathbf{p_i} \in \mathcal{P} : \mathbf{p_i} \in \partial B_r(\mathbf{p})$; thus, observe that $g(\mathbf{p_i}, \mathbf{n_i}, \frac{1}{2r}) = (\mathbf{p}, r)$ as expected, since otherwise the ball would . Notice that such $\mathbf{p_i}$ must exist: we know that $\exists \mathbf{x} \in (\partial B_r(\mathbf{p}) \cap \partial \mathcal{S})$, this in turn implies that $\exists i \in [N]$ such that the associated ball to $\mathbf{p_i}$ has $\mathbf{x}$ in its boundary, as it is determined by the NCH SDF. □

Remark. It is impossible to establish a bijection because two different points in the NCH may indeed transform to the same MAT. For example, a noise-free sample of a sphere

will have the same radius for every point. In this sense, the NCH is a more verbose representation than the MAT; however, it can always be pruned to be as small as the MAT without suffering from any changes in the shape being represented. It is very uncommon to suffer from this problem when looking at every day objects, so the deduplication step can be skipped completely in practice.

What this lemma tells us is that the NCH covers the entire MAT, proving that it is a viable representation for general shapes. Indeed, this proof also implies that many theorems that are true for the MAT are also true for the constrained NCH. What is missing is a reason to prefer the extension it over the MAT; the reason comes directly from the fact that planes are allowed.

It is well known that the MAT has problems representing sharp edges. From a theoretical standpoint, this is shown through the fact that the Medial Axis needs to include limit points in order to include sharp edges [GK04]. In practice, this means that there can't be an arbitrarily accurate reconstruction of sharp edges unless more and more points are added, which follows from the fact that $\mathbf{LFS}(\mathbf{w}) \to 0$ as $\mathbf{w} \in \partial\mathcal{S}$ gets closer to an edge point and, furthermore, that any implementation attempting to represent such surfaces needs to use an infinite number of Medial Atoms. Practical applications that use the MAT usually either:

- Do not handle this case. Since most methods produce one Medial Atom per sample in the input cloud, this also implies that arbitrary precision in the reconstruction of sharp edges requires denser samples in the corners, this is well handled when asking for ϵ-samples.

- Wade around this limitation by artificially generating such needed balls [SKS12]. Of course, since the representation in a computer is finite, this also implies that the approximation is only as good as the amount of balls that are generated.

One of the significant benefits of the NCH is that it does not suffer from such limitation: it is naturally able to represent sharp edges, simply by fitting planes instead of spheres. For example, a cube may be represented with only one sample per side, with normals pointing accordingly outwards.

Furthermore, the MAT has long been known to produce poor reconstructions when few sample points are available, such as in fig. 3.5. As we will see later on, performing isoextraction on the Symmetric NCH gives much better quality results, and particularly so for small point clouds, which fixes this problem.

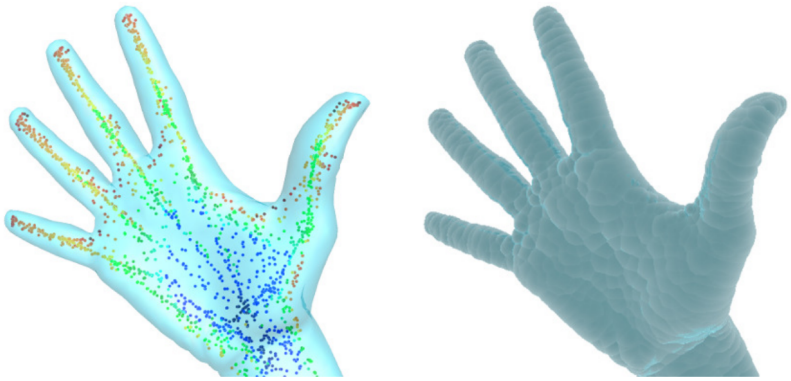


Fig. 3.5: A hand model and some of its Medial Atoms in the left side, the reconstruction of the hand as a union of balls in the right side.

Source: Delamé et al. [DRF12]

Concretely, a good NCH representation can potentially be both more accurate and compact than the MAT, key features when designing computer programs that need to scale. Furthermore, much of the literature developed for the MAT can be adjusted to the NCH, or at least applies when assuming that no plane fits are involved.

This last assumption is reasonable usually in the Inner NCH, as any watertight shape will have at least one point in the direction of its normal when it is sufficiently sampled. When the Outer NCH is involved, it is likely for fits to be planes, especially in examples such as spheres and cubes; in this case, the formalism has not yet been developed. Such a theoretical undertaking is beyond the scope of this thesis, which is why we limit our theoretical analysis to the Inner case.

3.3.4 Relationship with the Voronoi and Power Diagrams

Here, we will use the definitions from Amenta et al. [ACK01], although they are standard and come from other sources as well.

Definition 3.3.9 (Power Distance)**.** Let $B_1 = B_{r_1}(\mathbf{c_1})$ and $B_2 = B_{r_2}(\mathbf{c_2})$ be two balls, with r_1 and r_2 as their radii, and c_1 and c_2 as their centers, then:

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

It is easy to verify that this is a proper distance function.

Definition 3.3.10 (Power Diagram)**.** The power diagram $\text{Pow}(\mathcal{B})$ of a set of balls is the weighted Voronoi diagram which assigns an (unweighted) point $\mathbf{x}$ in space to the cell of the ball B which minimizes $d_{\text{pow}}(\mathcal{B}, B_0(\mathbf{x}))$.

Lemma 3.3.5. *Let $(\mathcal{P}, \mathcal{N})$ be a oriented point cloud, with ρ a finite set of the same size as $\mathcal{P}$ such that $\rho_i > 0$ for all $\mathbf{p_i}$. Also let $\mathcal{B} = \{B_{r_j}(\mathbf{p_j} + \mathbf{r_j n_j}) | (\mathbf{p_j}, \mathbf{n_j}, \rho_j) \in \mathbf{NCHA}(\mathcal{P}, \mathcal{N}, \rho)\}$, denoting each one B_j, and $B_x = B_0(\mathbf{x}) = \{\mathbf{x}\}$. Then:*

$$f_j(\mathbf{x}) \geq f_i(\mathbf{x}) \iff \frac{d_{pow}(B_j, B_x)}{r_j} \leq \frac{d_{pow}(B_i, B_x)}{r_i}$$

Proof. Recall from lemma 3.3.1 that:

$$\begin{aligned} f_j(\mathbf{x}) &= \frac{1}{2r_j}\{r_j^2 - ||\mathbf{x} - (\mathbf{p_j} + r_j\mathbf{n_j})||^2\} \\ &= \frac{-d_{\text{pow}}(B_j, B_x)}{2r_j} \end{aligned}$$

Where the second equation follows directly from definition 3.3.9. By simple replacement:

$$\begin{aligned} f_j(\mathbf{x}) = \frac{-d_{\text{pow}}(B_j, B_x)}{2r_j} &\geq \frac{-d_{\text{pow}}(B_i, B_x)}{2r_i} = f_i(x) \\ \frac{d_{\text{pow}}(B_j, B_x)}{r_j} &\leq \frac{d_{\text{pow}}(B_i, B_x)}{r_i} \end{aligned}$$

□

Corollary 3.3.2. *Using the same definitions as in lemma 3.3.5, notice that:*

$$f_j(\mathbf{x}) = f_i(\mathbf{x}) \iff \frac{d_{pow}(\mathbf{x}, B_j)}{r_j} = \frac{d_{pow}(\mathbf{x}, B_i)}{r_i}$$

Corollary 3.3.3. *The set of points $\{\mathbf{x} | f_i(\mathbf{x}) = f_j(\mathbf{x})\}$ is the set of roots to the quadratic polynomial given by the equation from corollary 3.3.2.*

The proof from lemma 3.3.5 is important because it basically establishes what is called a Multiplicatively Weighted Voronoi Diagram, see fig. 3.6 for an example. Suffice it to say, this mathematical object has been studied on its own and there exist algorithms for computing it, which indeed means it might be possible to use Voronoi-based methods for determining the areas where each function dominates, with some additional work required for planes, and thus optimize the evaluation of the SDF.

Voronoi methods are an extense research area by themselves, and thus further analysis will be considered out of scope in this thesis due to time constraints. For the interested reader, [Dob] may provide a good starting point.

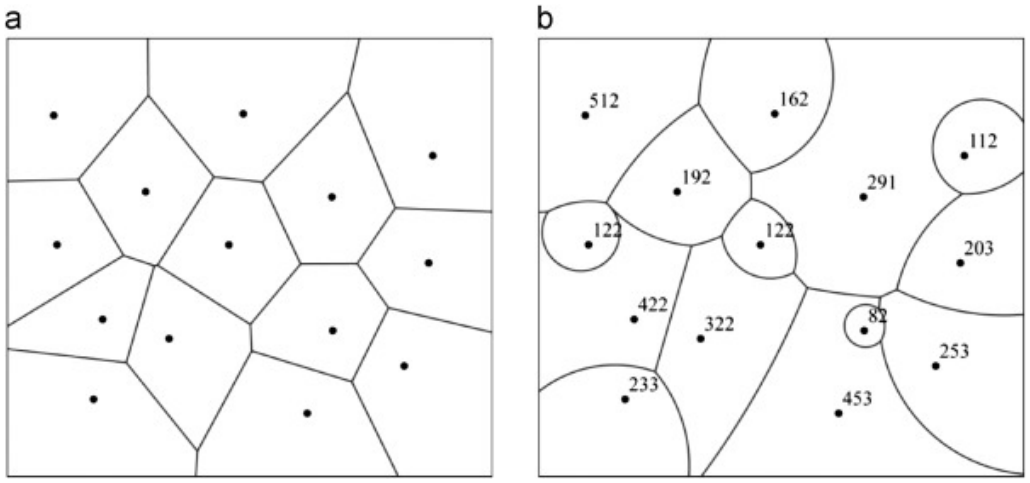


Fig. 3.6: (a) A Voronoi Diagram for a set of points. (b) A Multiplicatively Weighted Voronoi Diagram.

Source: Cărbunar et al. [Căr+06]

3.4 Practical Algorithms

Up to this point, the presentation has been mostly theoretical in nature: we have presented the MAT and NCH as mathematical objects, and shown some of their properties; as well as introduced several basic concepts in 3D surface reconstruction, such as SDFs. From here on, the rest of this thesis is focused on showing practical algorithms for surface reconstruction using the NCH, and studying both its theoretical properties and experimental behavior.

3.4.1 Naïve Non-Convex Hull

The Naïve Non-Convex Hull (NNCH) algorithm is the estimation of ρ in the way that was described in section 3.3.1. It is important to remark that this is just one way to estimate ρ; it does have the nice properties described above, but its rigidness means that it is not robust to noise.

In terms of costs, computing ρ_i is an $\Theta(N)$ operation, which means that computing ρ is an $\Theta(N^2)$ operation, as seen in algorithm 3.1. Notice that the space complexity is $\Theta(N)$, and the time complexity of computing a value of the SDF is $\Theta(N)$, as seen in algorithm 3.2. Since we are working in $\mathbb{R}^3$, this brings the cost of Marching Cubes to $\Theta((K+1)^3 N)$ for a resolution of $\mathcal{O}(K)$ in all 3 directions; of course, in $\mathbb{R}^2$ we would have a cost of $\Theta((K+1)^2 N)$ under the same circumstances.

When working with any practical point clouds, this is prohibitively expensive, as most

Algorithm 3.1 Estimate the NCH using the NNCH algorithm

for $i \in 1 \dots N$ **do**
  $\rho_i \leftarrow 0$
  for $j \in 1 \dots N$ **do**
    if $i \neq j$ **then**
      $a \leftarrow \langle \mathbf{n_i}, \mathbf{p_j} - \mathbf{p_i} \rangle$
      $b \leftarrow ||\mathbf{p_j} - \mathbf{p_i}||^2$
      if $a > \rho_i b$ **then**
        $\rho_i \leftarrow \frac{a}{b}$ {Happens iff $\mathbf{p_j} \in J_i$ and it shrinks the Medial Atom}
      end if
    end if
  end for
end for
return $\{\rho_1, \dots, \rho_N\}$

Algorithm 3.2 Obtain the SDF value for a given point using the NCH SDF

$f_x \leftarrow -\infty$
for $i \in 1 \dots N$ **do**
  $a \leftarrow \langle \mathbf{n_i}, \mathbf{x} - \mathbf{p_i} \rangle - \rho_i ||\mathbf{x} - \mathbf{p_i}||^2$
  if $a > f_x$ **then**
    $f_x \leftarrow a$
  end if
end for
return f_x

of them have way over 100k points. For point clouds with 800k points, for example, it is impossible to run even the estimation procedure.

There are a few more properties that will be very convenient when manipulating the NCH, and are a good algorithm sanity check:

Lemma 3.4.1. *The NNCH algorithm is rotation invariant.*

Proof. Let $\mathbf{p'_i} = Q\mathbf{p_i}$ and $\mathbf{n'_i} = Q\mathbf{n_i}$ with Q an orthogonal matrix, and f', f'_i, ρ'_i defined as above. Then, if we look at the ρ'_i estimation:

$$\begin{aligned}\rho'_i &= \max_{j \in J'_i} \frac{\mathbf{n'_i}^t(\mathbf{p'_j} - \mathbf{p'_i})}{||\mathbf{p'_j} - \mathbf{p'_i}||^2} \\ &= \max_{j \in J'_i} \frac{\mathbf{n_i}^t Q^t Q(\mathbf{p_j} - \mathbf{p_i})}{||Q(\mathbf{p_j} - \mathbf{p_i})||^2} \\ &= \max_{j \in J'_i} \frac{\mathbf{n_i}^t(\mathbf{p_j} - \mathbf{p_i})}{||\mathbf{p_j} - \mathbf{p_i}||^2}\end{aligned}$$

Furthermore, if we look at the set J'_i:

$$\begin{aligned}J'_i &= \{j : \mathbf{n'_i}^t(\mathbf{p'_j} - \mathbf{p'_i}) > 0\} \\ &= \{j : \mathbf{n_i}^t Q^t Q(\mathbf{p_j} - \mathbf{p_i}) > 0\} \\ &= \{j : \mathbf{n_i}^t(\mathbf{p_j} - \mathbf{p_i}) > 0\} \\ &= J_i\end{aligned}$$

Which implies that $\rho'_i = \rho_i$. Finally, $f'_i(Q\mathbf{x})$:

$$\begin{aligned}f'_i(Q\mathbf{x}) &= \mathbf{n'_i}^t(Q\mathbf{x} - \mathbf{p'_i}) - \rho'_i||Q\mathbf{x} - \mathbf{p'_i}||^2 \\ &= \mathbf{n_i}^t Q^t Q(\mathbf{x} - \mathbf{p_i}) - \rho_i||Q(\mathbf{x} - \mathbf{p_i})||^2 \\ &= \mathbf{n_i}^t(\mathbf{x} - \mathbf{p_i}) - \rho_i||\mathbf{x} - \mathbf{p_i}||^2 \\ &= f_i(\mathbf{x})\end{aligned}$$

Thus, when we evaluate on a rotated point $Q\mathbf{x}$, we get the guarantee that:

$$f'(Q\mathbf{x}) = f(\mathbf{x})$$

□

Lemma 3.4.2. *The NNCH algorithm is translation invariant.*

Proof. The proof strategy here is all too similar to the previous case, the difference being that now our points are changed to $\mathbf{p_i'} = \mathbf{p_i} + \mathbf{c}$, and $\mathbf{n_i'} = \mathbf{n_i}$. In this case, $\mathbf{p_j'} - \mathbf{p_i'} = \mathbf{p_j} - \mathbf{p_i}$ because the constant cancels out; this trivially implies that $J_i' = J_i$ and $\rho_i' = \rho_i$. Finally, the new guarantee that we are going to get is that:

$$f_i'(\mathbf{x} + \mathbf{c}) = f_i(\mathbf{x})$$

Derived out of $\mathbf{c}$ canceling out in the sum, and finally:

$$f'(\mathbf{x} + \mathbf{c}) = f(\mathbf{x})$$

□

Lemma 3.4.3. *The NNCH algorithm is not scale invariant, but level sets are predictable.*

Proof. Proceeding in the same way as the other cases, we will take $\mathbf{p_i'} = c\ \mathbf{p_i}\ \forall \mathbf{p_i} \in \mathcal{P}$, $c > 0, c \in \mathbb{R}$ this time. In doing so, $J_i' = J_i$ trivially as the sign is not changed. Furthermore,

$$\rho_i' = \frac{1}{c}\rho_i$$

Which leads to $f_i'(c\mathbf{x}) = cf_i(\mathbf{x})$. When we replace in $f(\mathbf{x})$:

$$\begin{aligned} f'(c\mathbf{x}) &= \max_{1 \le i \le N} f_i'(c\mathbf{x}) \\ &= \max_{1 \le i \le N} cf_i(\mathbf{x}) \\ &= c \max_{1 \le i \le N} f_i(\mathbf{x}) \\ &= cf(\mathbf{x}) \end{aligned}$$

Indeed, when $f(\mathbf{x}) = 0 \iff f'(c\mathbf{x}) = 0$, this means we can always scale appropriately to retain the level set, even if we are working with a smaller-scale model in the first place. □

Remark. It is also possible to mirror the surface, by setting $\mathbf{p_i'} = -\mathbf{p_i}$, $\mathbf{n_i'} = -\mathbf{n_i}$. ρ will stay the same. It is straightforward to prove that $f(-\mathbf{x}) = f(\mathbf{x})$.

These properties are very useful when there is interest in manipulating the shape before or after we have fit the NCH. For example, they can be used to scale the point cloud to fit inside a bounding box, estimate the NCH with the scaled points, and then turn back the representation into the original model before isoextraction.

3.4.2 Shrinking Ball

The Shrinking Ball (SB) algorithm is an algorithm for estimating the MAT, published by Ma et al. [MBC12], with follow-up work by Jalba et al. [JKT13], as well as Peters [Pet18] [Pet14d] [Pet14a] and Lam [Lam16]. The core idea itself is very similar to what is done in NCH, and has also been spotted in unrelated works in one way or another [SKS12] [BS12].

The algorithm is based on two facts: first, that given a point $\mathbf{x} \in \partial\mathcal{S}$, there must exist a $B = B_r(\mathbf{p}) \in \mathbf{MAT}(\mathcal{S})$ such that $\mathbf{x} \in \partial B$ and the normal of the ball at $\mathbf{x}$ must be the normal of $\mathbf{x}$ at $\partial\mathcal{S}$ (i.e., it ought to agree with the surface); and second, that there must be another $\mathbf{y} \in \partial\mathcal{S}$ that is also in ∂B, since by definition any Medial Atom (MA) must be maximally contained in the surface [She95; CCM97].

The algorithm, in a very similar format as in the original paper, can be seen in algorithm 3.3 and works as follows. Take an oriented input point cloud $(\mathcal{P},\mathcal{N})$ sampled from $\partial\mathcal{S}$, and imagine then that you want to find a finite approximation to $\mathbf{MAT}(\mathcal{S})$, one thing to do is fit one ball per point $\mathbf{p_j} \in \mathcal{P}$, using the aforementioned ideas: first generate a ball B such that $\mathbf{p_j}$ is at its border and the normal matches accordingly; then pick any other point $\mathbf{p_i} \in (\mathcal{P} \cap \text{int}(B))$[1], and compute a new maximal ball with $\mathbf{p_i}$ in the border. Due to the fact that the ball must be maximally contained, we can repeat the last process until the intersection is empty (i.e, the ball cannot shrink any more), which defines a sequence of progressively smaller balls $\mathcal{B}_j = \{B_{r_{ji}}(\mathbf{c_{ji}})\}_{i\in\mathbb{N}}$ that converges to a ball $B_{r_j}(\mathbf{c_j})$. If the sample is noise-free and sufficiently dense, doing this will result in a set of maximal balls $\mathcal{B} = \{B_{r_1}(\mathbf{c_1}), \ldots, B_{r_N}(\mathbf{c_N})\} \subset \mathbf{MAT}(\mathcal{S})$. An example algorithm trace can be seen in fig. 3.7.

Indeed, the NNCH algorithm from section 3.4.1 is doing exactly this, except that the initial "ball" is actually of infinite radius (i.e. a plane), and instead of iteratively picking points, every single point is considered. Seen this way, the SB algorithm is a clear optimization; however, there are two key issues with this idea.

To start with this procedure, we first need an initial radius for the current point (this is exactly what the InitialRadius auxiliary function does); since there is no way to select the initial radius optimally, heuristics are used, and hence the AdjustRadiusHeuristic auxiliary function: it ought to adjust the heuristic for the next point to be processed. Thus, the first issue is that it is unclear what the initial radius should be, as it can not be infinite (i.e. a plane): if the initial ball is too small, then it is not a Medial Atom, and indeed $\mathcal{B}$ will not cover the entire surface; if it is too large, it will protrude from the surface and convergence will be slow. The optimal initial value for a given $\mathbf{p_j}$ is by definition $\mathbf{LFS}(\mathbf{p_j})$,

[1] Here $\text{int}(B)$ denotes the interior of the set, using the standard definition

Algorithm 3.3 Compute the MAT using the SB algorithm by Ma et al. [MBC12], as derived from the original code by Peters [Pet18]

for $i \in 1 \dots N$ **do**
  $r_i \leftarrow \text{InitialRadius}(i)$
  for $t \in 1 \dots t_{\max}$ **do**
    $j \leftarrow \text{NearestNeighbor}(\mathbf{p_i} + r_i \mathbf{n_i}, \mathbf{p_i})$ {The search excludes $\mathbf{p_i}$}
    $r_i' \leftarrow \text{ComputeTangentSphereRadius}(\mathbf{p_i}, \mathbf{p_j})$ {As in NNCH: $\frac{||\mathbf{p_j}-\mathbf{p_i}||^2}{2\langle \mathbf{n_i}, \mathbf{p_j}-\mathbf{p_i}\rangle}$}
    $\text{swap}(r_i, r_i')$
    if $|r_i - r_i'| < \delta$ **then**
      break {The algorithm has converged}
    end if
  end for
  $\text{AdjustRadiusHeuristic}(i)$
end for
return $\{r_1, \dots, r_N\}$

and thus $\sup_{\mathbf{p}\in\mathcal{S}} \mathbf{LFS}(\mathbf{p})$ would be a reasonable initial value for every point; however, the LFS is of course unavailable, so we are forced to approximate it with heuristics.

Ma et al. [MBC12] proposed a heuristic that sets the initial radius for a ball as follows:

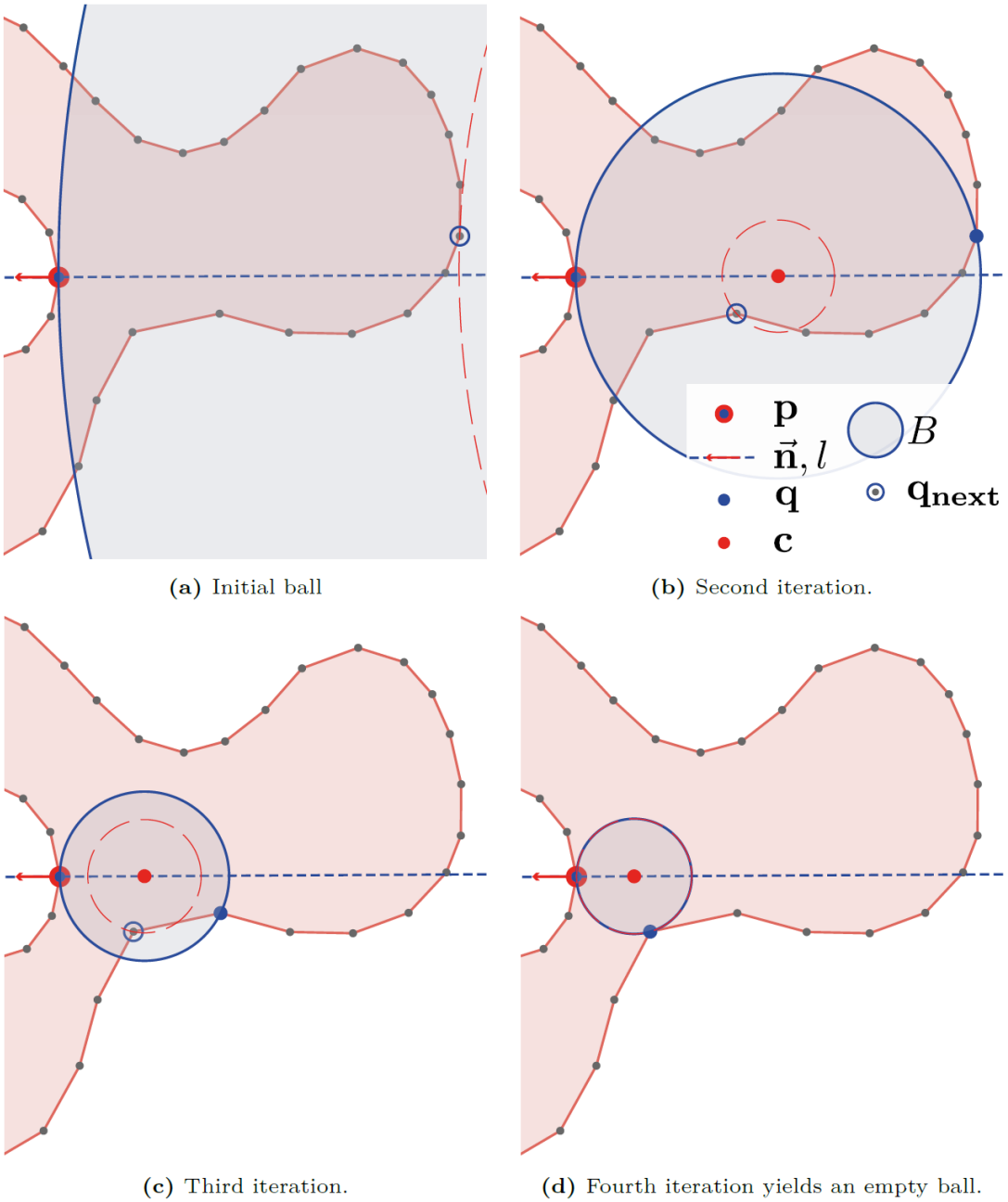
1. At the first point, just choose any other point at random and use it as border to compute the initial radius.

2. For all following points, set it to the radius of a ball, in the proper direction of the normal for the point, which keeps both the current and previously fit point at the border.

The heuristic reduces the number of shrinking steps to an average of 2 instead of 8 -according to their experiments-, but requires processing the points by following an in-order iteration of a KD-Tree containing the cloud [2]. This requirement directly translates to synchronization needs in parallel implementations, which makes them very complicated, and also significantly increases implementation complexity.

From a theoretical standpoint, it exploits the fact that the next point should always keep the current point outside of its MA, which follows from the maximality requirement. However, this has a crucial problem, which is that such an atom does not necessarily exist: the previous point could be in the opposite direction of the current point's normal, which

[2] If this doesn't happen, then points may be too far, and the LFS may have changed too much



(a) Initial ball

(b) Second iteration.

(c) Third iteration.

(d) Fourth iteration yields an empty ball.

Fig. 3.7: An example of an SB run, where **p** is the point being fit, **n** its normal, **c** is the center of the ball fit, **q** the point that determines the ball, and $\mathbf{q}_{\text{next}}$ is the nearest point to the current center (i.e. the point to determine the next ball).

Source: Peters [Pet18]

would require a negative radius. The authors did not elaborate on this problem in the paper, and it is a serious one, especially when dealing with noisy point clouds.

Jalba et al. [JKT13] presented a parallelization-friendly approach to initial radius es-

timation, which consists of dividing the point cloud into equal-sized chunks -without any point ordering- and processing one chunk per thread. There is a single global initial radius with a default value, which is updated every time a point's final radius is found by using a moving average. Concretely, they estimate the initial radius to a number similar to the average LFS for the sample.

Although the heuristic beats the method of [MBC12] in performance, mainly because it manages to move more processing into the GPU, it is important to note that it is unsound from a theoretical point of view: a priori, the average LFS can be severely smaller than the actual LFS of any given point -even more so if the points are far apart-, which translates into smaller balls than required (i.e. holes in the surface). They do not comment on this, and we could not find their source code for further experimentation or inspection.

Peters [Pet18] [Pet14d] [Pet14a] and Lam [Lam16] use an user-supplied maximum radius for all points. It is a reasonable way to avoid the problem, as we usually have an estimate of the maximum two points can be apart from each other, which is an upper bound to the LFS.

Alternatively, we can estimate an upper bound to the maximum distance between any two points in linear time by obtaining a bounding box, and this is guaranteed to be larger than the local feature size at any given sample, making it correct to use as a constant initial radius.

The second issue is that we need to find $\mathbf{p_i}$, the point to move the border of the current Medial Atom estimate, but we have not specified how to do so; the only thing we have said about $\mathbf{p_i}$ is that $\mathbf{p_i} \in (\mathcal{P} \cap \text{int}(B))$. The key idea here is that we can add another condition to make it easier to find, basically by constraining further, so that $\mathbf{p_i}$ is not just in the interior, but also the closest possible to B's center; in this case, it is easy to find such a point by doing a nearest neighbor query on B's center, with maximum radius given by B as well (as in line 4 of the algorithm).

There are multiple data structures that support Nearest Neighbor searches in 3D; one of the most common choices is to use a KD-Tree. These trees partition space into regions determined by hyper-planes that best separate a given set of points, as shown in fig. 3.8; see Berg et al. [Ber+08] for a detailed description. Although the most common usage for KD-Trees is to do range searches, they are also often used for Nearest Neighbor and Radius Searches. All known implementations of SB thus far use KD-Trees, including ours.

In terms of time complexity, it is important to point out that all SB variants are $\mathcal{O}(N^2)$, as we have to process each point, and the nearest neighbor search may take $\mathcal{O}(N)$ in the worst case. Under the assumption that the data structure used for lookup is a KD-Tree,

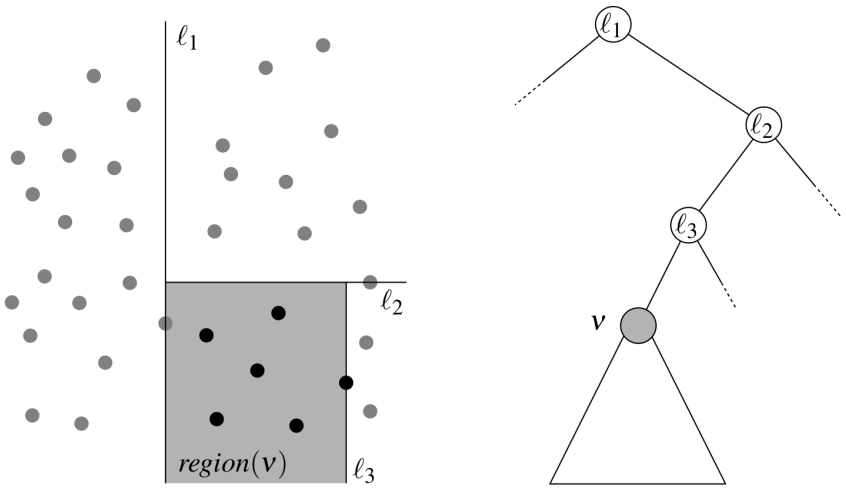


Fig. 3.8: Points in $\mathbb{R}^2$, and a partially constructed KD-Tree. This picture shows the correspondence between each node in the tree and the region it represents. Searches are carried out by checking which side of the plane the point is at.

Source: Berg et al. [Ber+08]

the average complexity of the search is $\mathcal{O}(\log N)$, which implies that the average time complexity is $\mathcal{O}(N \log N)$, a significant improvement from NNCH's $\Theta(N^2)$. Furthermore, the algorithm has a spatial complexity of $\mathcal{O}(N)$, since that is how much the KD-Tree takes and we only require to store the corresponding radii.

Notice that the number of iterations in the refinement have not entered in our analysis, this is because this is a constant in practice. Moreover, in practice the number of iterations has been less than 10 for all models tested in this work, and can be systematically reduced when using a good radius initialization heuristic.

3.4.3 Denoising Shrinking Ball

Peters [Pet18; Pet14d; Pet14a] developed a method based on the θ-SMA. The θ-SMA is a variation of the Medial Axis Transform developed by Foskey et al. [FLM03] which consists of the subset of Medial Atoms whose maximum separation angles between tangent points are greater than θ. More formally, if $S(\mathbf{x})$ is taken as in fig. 3.10, then the θ-SMA is the subset of the MAT such that $S(\mathbf{x}) > \theta$ for any given center of a Medial Atom.

In fig. 3.9, the effect of increasing θ can be seen: as the minimum required separation angle gets larger, more and more atoms are removed from the MAT, generating a simplified representation. Notice in particular the evolution of the Triceratops model: with $\theta = 15^\circ$, the horns have already been removed, and by $\theta = 60^\circ$, most of the features shown in the

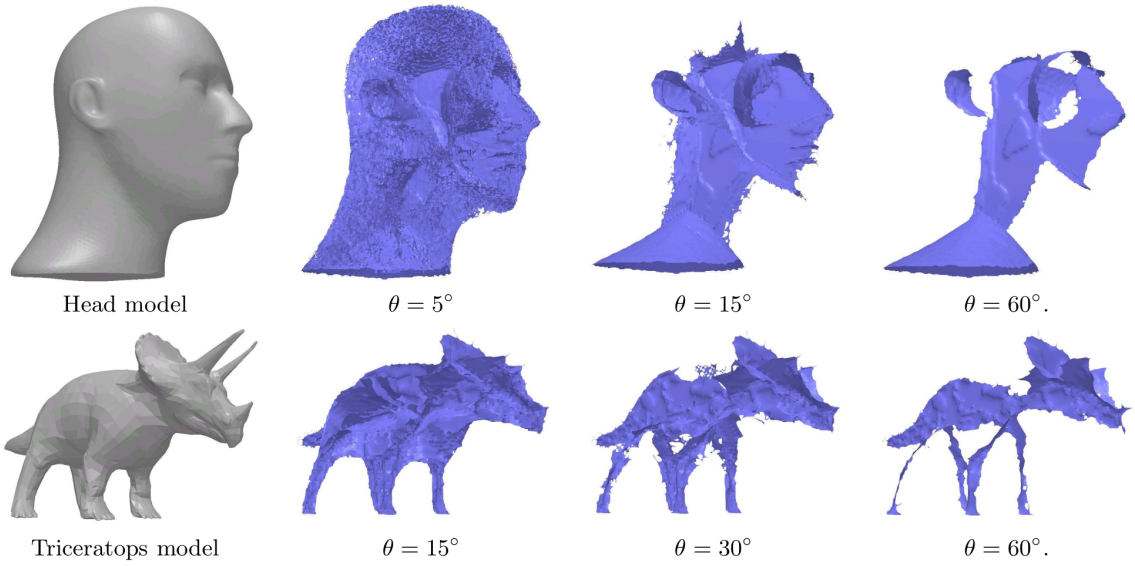


Fig. 3.9: Example of the different MATs when using the angle filtering technique.

Source: Foskey et al. [FLM03]

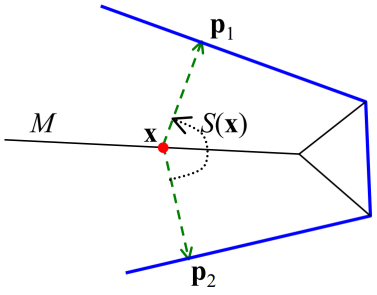


Fig. 3.10: The separation angle $S(\mathbf{x}) = \max_{\mathbf{p_1},\mathbf{p_2} \in N(\mathbf{x})} \sphericalangle \mathbf{p_1 x p_2}$, and $N(\mathbf{x})$ represents the set of points in $\partial\mathcal{S}$ closest to $\mathbf{x}$ than to any other point in the medial axis

Source: Foskey et al. [FLM03]

MAT have been highly filtered out.

The idea comes from the Voronoi filtering approaches used to compute the MAT, and in particular from the work by Dey and Zhao [DZ02] and Amenta et al. [ACK01], which linked small separation angles to MAT instability. Seen this way, the heuristic is an attempt to remove all MAT instability and it can be proven that, as $\theta \to 0$, the θ-SMA converges to the MAT.

Peters [Pet18] adapted the heuristic developed by Dey and Zhao [DZ02] to work within the SB algorithm by stopping the shrinking process as soon as a ball with a separation angle too small is found, as seen in algorithm 3.4 and shown in fig. 3.11. In this way, no points are removed from the cloud (thus maximizing point efficiency, crucial in LiDAR datasets such as Peters') and instability is avoided.

The issue with this approach is that the process itself may generate errors by using balls with too large separation angles; furthermore, the fact that the current ball's separation angle is small does not imply that a future ball will still have a small separation angle, so stopping the shrinking may in fact prevent the finding of good Medial Atoms. Indeed, this causes the elimination of small features in the surface; while for LiDAR point clouds it is less of a problem, when doing surface reconstruction that heavily depends on the surface itself.

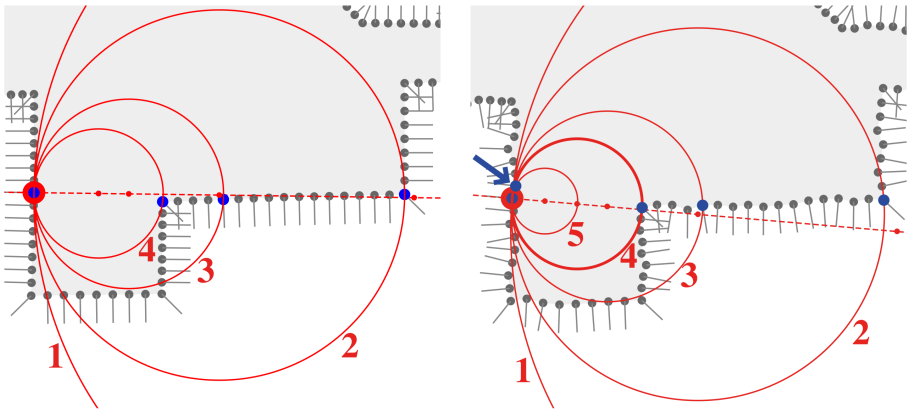


Fig. 3.11: The left image shows a shrinking ball run as usual, while the right one has a noisy point. Using the heuristic prevents the right situation by early stopping.

Source: Peters [Pet18]

Another problem is that the hyper-parameter θ is model-dependant, and no way to estimate a good value has been developed to the best of our knowledge. Moreover, too small values will lead to not removing the noise, and too large values will result in severe quality loss by effectively removing most features in the surface.

This last issue in particular implies that the reconstruction process must be run several times before the optimal θ is found. It is inconvenient enough that we have decided to leave the DSB algorithm as future work and not experiment with it.

Algorithm 3.4 Compute the MAT using the Denoising Shrinking Ball (DSB) algorithm by Peters [Pet18], as seen in the open source implementation (linked).

for $i \in 1 \dots N$ **do**
  $r_i \leftarrow \text{InitialRadius}(i)$
  for $t \in 1 \dots t_{\max}$ **do**
    $j \leftarrow \text{NearestNeighbor}(\mathbf{p_i} + r_i\mathbf{n_i}, \mathbf{p_i})$ {The search excludes $\mathbf{p_i}$}
    $r'_i \leftarrow \text{ComputeTangentSphereRadius}(\mathbf{p_i}, \mathbf{p_j})$ {As in NNCH: $\frac{||\mathbf{p_j}-\mathbf{p_i}||^2}{2\langle \mathbf{n_i}, \mathbf{p_j}-\mathbf{p_i}\rangle}$}
    if $|r_i - r'_i| < \delta$ **then**
      $r_i \leftarrow r'_i$
      break {The algorithm has converged}
    end if
    $\mathbf{c'_i} \leftarrow \mathbf{p_i} + r'_i\mathbf{n_i}$ {The center at the next iteration}
    $\theta \leftarrow \text{AngleBetween}(\mathbf{p_i} - \mathbf{c'_i}, \mathbf{p_j} - \mathbf{c'_i})$ {By the law of cosines: $\arccos \frac{\langle \mathbf{p_i}-\mathbf{c'_i}, \mathbf{p_j}-\mathbf{c'_i}\rangle}{||\mathbf{p_i}-\mathbf{c'_i}||||\mathbf{p_j}-\mathbf{c'_i}||}$}
    if $t = 0 \wedge \theta < \theta_{\text{planar}}$ **then**
      break
    end if
    if $t > 0 \wedge \theta < \theta_{\text{preserve}} \wedge r_i > ||\mathbf{p_j} - \mathbf{p_i}||^2$ **then**
      break
    end if
    $r_i \leftarrow r'_i$
  end for
  $\text{AdjustRadiusHeuristic}(i)$
end for
return $\{r_1, \dots, r_N\}$

4. ALGORITHMIC IMPROVEMENTS

4.1 Shrinking Planes

The SB algorithm as introduced in section 3.4.2 and section 3.4.3 has an error that was not corrected in the implementation or thesis by Peters [Pet18], nor mentioned in the original paper by Ma et al. [MBC12].

When looking for a nearest neighbor of the center point, there is a priori no restriction that it be in the appropriate side of the normal of the point (that it be in J_i, in NCH terminology). Hence, the radius computation would return a negative radius, and thus computing the opposite Medial Atom, thereby crippling the reconstruction process.

It makes sense not to check for this: indeed, the point being processed should always be nearer than any point in the other side of the plane determined by the point and the normal. However, in the PCL implementation of a KDTree, it happens in practice that a point in the other side of the normal is returned.

In particular, this was found doing reconstructions of the Outer MAT, which is usually more prone to edge cases. A corrected version is shown in algorithm 4.1. This was a particularly elusive error, which would manifest in warts in places there should not be any, and sometimes the disappearance of surface parts; an example will be shown in section 4.3.1.

The SB algorithm works for finding a MAT approximation. A natural extension is to modify it to find the NCH of a set of points. There are many different ways to do so, the chosen one is to mimic what the NNCH algorithm does: begin with a plane, and shrink from there on. Since it is not actually possible to use the SB algorithm with a plane as an initial guess (because there is no center to the Medial Atom being fit), a reasonable adaption is to abuse the radius initialization.

Definition 4.1.1 (Radius Initialization Function)**.** A function $h : [N] \to \mathbb{R}_{>0}$ is a radius initialization function when

$$h(j) > \mathbf{LFS}(\mathbf{p_j}) \; \forall \mathbf{p_j} \in \mathcal{P}$$

Remember from section 3.4.2 that the initial estimate must never be smaller than $\mathbf{LFS}(\mathbf{p_j})$, because in such cases the ball will never shrink and artifacts will be generated inside the surface or the border will change. The additional condition that it is not exactly

Algorithm 4.1 Compute the MAT using the SB algorithm with the correctness change required. Added stop highlighted.

1: **for** $i \in 1 \dots N$ **do**
2: $r_i \leftarrow \text{InitialRadius}(i)$
3: **for** $t \in 1 \dots t_{\max}$ **do**
4: $j \leftarrow \text{NearestNeighbor}(\mathbf{p_i} + r_i\mathbf{n_i}, \mathbf{p_i})$ {The search excludes $\mathbf{p_i}$}
5: $r'_i \leftarrow \text{ComputeTangentSphereRadius}(\mathbf{p_i}, \mathbf{p_j})$
6: **if** $r'_i < 0$ **then**
7: **break** {The nearest point may be in the wrong direction of the normal}
8: **end if**
9: $\text{swap}(r_i, r'_i)$
10: **if** $|r_i - r'_i| < \delta$ **then**
11: **break** {The algorithm has converged}
12: **end if**
13: **end for**
14: $\text{AdjustRadiusHeuristic}(i)$
15: **end for**
16: **return** $\{r_1, \dots, r_N\}$

the $\mathbf{LFS}(\mathbf{p_j})$ is equivalent to non-optimality: it should never happen that the initial radius be exactly the final radius.

If a proper Radius Initialization Function is given, there has to be at least one adjustment per point in any sufficiently dense point cloud, as we are in the situation pictured in fig. 4.1. There is, however, an exception for points that ought to be planes instead: in these cases, there is simply no point that can be used to compute a new radius, and hence none will be found, causing a break from the iterative process in the very first iteration, as in fig. 4.2. Hence, we can check if the current radius is still the same as the initial one, and if so fit a plane. A pseudo-code can be seen in algorithm 4.2.

Notice that if the radius was less than or equal to the LFS for any given point, then the initial radius would not be modified at all, and thus we would fit a plane as shown in the pseudo-code. This would of course be incorrect.

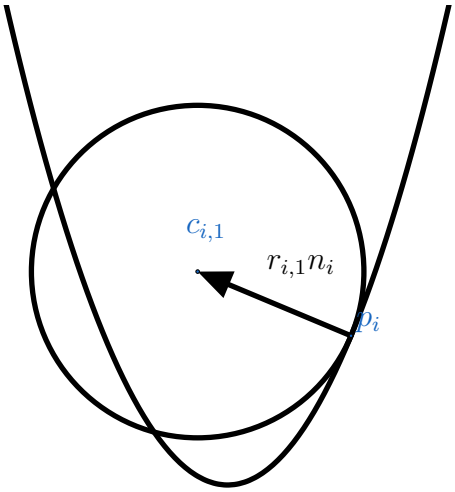


Fig. 4.1: This is a normal Inner NCH fitting situation with a good Radius Initialization Function. Since the radius is slightly larger than it needs to be, any sufficiently sampled point cloud will result in at least one radius adjustment.

Notice that finding a Radius Initialization Function is easy: any strict upper bound on the LFS works, and in particular the maximum distance between any two points plus a small epsilon will always be a -constant- initial guess that satisfies the conditions.

The algorithm as presented already works for most shapes in the tested data, but fails subtly in others. The cause was elusive to find, but easy to understand: it is likely to break the loop due to fake convergence issues, since it often happens that the radiuses do not change severely from one iteration to the next. One fix for this issue is to enforce the maximality of the Medial Atom. The key insight is that after the inner loop has finished

Algorithm 4.2 Compute the NCH using the SB variant. Key change highlighted.

1: **for** $i \in 1 \dots N$ **do**
2: $r_i \leftarrow \text{InitialRadius}(i)$
3: **for** $t \in 1 \dots t_{\max}$ **do**
4: $j \leftarrow \text{NearestNeighbor}(\mathbf{p_i} + r_i\mathbf{n_i}, \mathbf{p_i})$ {The search excludes $\mathbf{p_i}$}
5: $r'_i \leftarrow \text{ComputeTangentSphereRadius}(\mathbf{p_i}, \mathbf{p_j})$
6: **if** $r'_i < 0$ **then**
7: **break** {The nearest point may be in the wrong direction of the normal}
8: **end if**
9: $\text{swap}(r_i, r'_i)$
10: **if** $|r_i - r'_i| < \delta$ **then**
11: **break** {The algorithm has converged}
12: **end if**
13: **end for**
14: **if** $r_i \geq \text{InitialRadius}(i)$ **then**
15: $r_i \leftarrow \infty$ {Fit a plane in the normal's direction}
16: **end if**
17: $\rho_i \leftarrow \frac{1}{2r_i}$ {Convert into the NCH}
18: $\text{AdjustRadiusHeuristic}(i)$
19: **end for**
20: **return** $\{\rho_1, \dots, \rho_N\}$

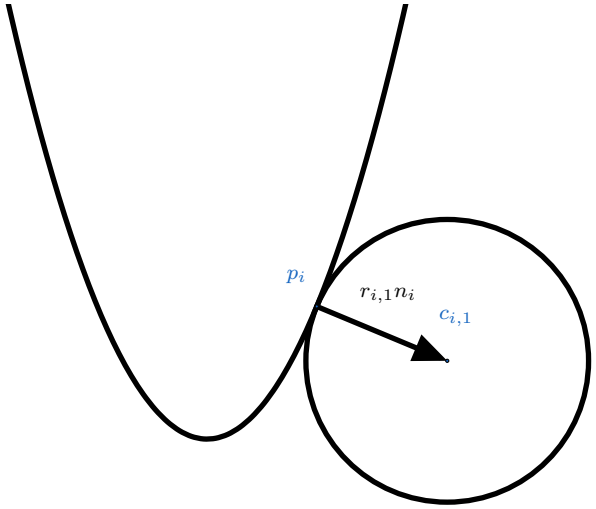


Fig. 4.2: The Outer NCH computation for the same situation as in fig. 4.1. There is no correct value for the LFS, so the loop will break in the first iteration, as it can't possibly find any point to shrink the ball with.

(line 13 of algorithm 4.2), we are in one of three conditions: supposed to have a plane -in which case it is impossible to shrink at all-, the ball has converged -i.e. it is maximal-, or the ball has shrunk partially -thus the estimate can be improved-.

In any case, we can run a radius search using a KD-Tree (which we already have for the nearest neighbor search), and shrink it so that all points found be outside the ball. In the first scenario, only $\mathbf{p_i}$ will be returned; in the second scenario, only border points will be found, and thus no shrinking will happen; in the third scenario, points will be found inside the ball, which will shrink it further into convergence, see algorithm 4.3 for the pseudo-code.

Line 17 All points inside the radius search are used for shrinking; this is due to points in the border sometimes being returned earlier than the rest, and because the actual expensive operation in the loop is the search (after it is done, the rest is just optimizing future operations).

Line 14 Multiple iterations of the loop are allowed. In practice, none of the shapes presented in this work took more than 2 iterations per point, and most of them only required one. Still, this allows for progressively shrinking the ball with different subsets.

It is important to point out that the radius search stage is not viable on its own as a

Algorithm 4.3 Compute the NCH using the SB variant with the final refinement stage

for $i \in 1 \dots N$ **do**
  $r_i \leftarrow$ InitialRadius(i) {Must be finite}
  for $t \in 1 \dots t_{\max}$ **do**
    $j \leftarrow$ NearestNeighbor$(\mathbf{p_i} + r_i\mathbf{n_i}, \mathbf{p_i})$ {The search excludes $\mathbf{p_i}$}
    $r_i' \leftarrow$ ComputeTangentSphereRadius$(\mathbf{p_i}, \mathbf{p_j})$
    if $r_i' < 0$ **then**
      break {The nearest point may be in the wrong direction of the normal}
    end if
    swap(r_i, r_i')
    if $|r_i - r_i'| < \delta$ **then**
      break {The algorithm has found a good approximation}
    end if
  end for
  for $y \in 1 \dots y_{\max}$ **do**
    $n_1, \dots \leftarrow$ RadiusSearch$(\mathbf{p_i} + r_i\mathbf{n_i}, r_i, k)$ {Up to $2 < k \leq \infty$ points inside the MA}
    changed $\leftarrow$ false
    for $n = n_1, \dots$ **do**
      if $n \neq i$ **then**
        $r_i' \leftarrow$ ComputeTangentSphereRadius$(\mathbf{p_i}, \mathbf{p_j})$
        if $r_i' < r_i$ **then**
          $r_i \leftarrow r_i'$
          changed $\leftarrow$ true
        end if
      end if
    end for
    if not changed **then**
      break {Early stop refinement}
    end if
  end for
  if $r_i \geq$ InitialRadius(i) **then**
    $r_i \leftarrow \infty$
  end if
  $\rho_i \leftarrow \frac{1}{2r_i}$
  AdjustRadiusHeuristic(i)
end for
return $\{\rho_1, \dots, \rho_N\}$

way to compute the MAT, although it would be a correct algorithm if $k = \infty$, basically by being equivalent to NNCH. When the radius estimate is too large compared to the true LFS, points will be heavily dispersed through the search area and processed in a random order. For example, all points near the border could be returned first, and this implies a huge sunk cost. If the radius search is constrained to be returned in descending sorted order from the center point, the search becomes very costly and the algorithm runs slowly in comparison. These problems imply that a good radius estimate is required before actually running it, and that is precisely what the first loop provides.

One possible problem with this algorithm as provided is the presence of noise, which may lead to over-shrinking. However, it should be straightforward to incorporate the DSB heuristic from section 3.4.3 into it. Such modifications will not be explored in this thesis due to time constraints, but are highly encouraged as future work.

On one hand, Ma et al. [MBC12] proved that given an ϵ-sample, the SB algorithm (corrected as is shown in algorithm 4.1) converges to a subset of the MAT, and furthermore it converges to the true MAT as $\epsilon \to 0$. This indeed proves that error-free reconstruction is possible with SB. On the other hand, Taubin [Tau13] established the same for the NNCH algorithm. The algorithms proposed are basically equivalent to both, so the approximation property derives directly from this: a plane is fit only when there are no points in J_i (which is exactly what NNCH does), and otherwise we are guaranteed to find the maximal ball because of the second loop. Of course, this is a priori only true when $k = \infty$, otherwise we may miss some point.

In terms of complexity, it is important to point out that algorithm 4.3 has a crucial difference with respect to the rest of the algorithms: it needs a radius search over the KD-Tree. The worst case time complexity of such a search is $\mathcal{O}(k \log N)$, where k is the number of points to be found, so that indeed turns the worst case complexity into $\mathcal{O}(kN \log N)$. In practice, the performance difference between the other SB variants and this one is negligible.

4.2 Evaluation Methodology and Comparison Criteria

Evaluation of surface reconstruction algorithms is a relatively underdeveloped topic. Most of the literature limits to reconstruct several point clouds and perform a qualitative analysis of the results, as well as publishing running times of the algorithms as a function of the number of points in the cloud.

Berger et al. [Ber+13] proposed a framework in which they start from implicit shapes given by an MPU [Oht+03] representation, sample a point cloud from the shape by mod-

elling the 3D scanner's process, run the reconstruction algorithm, and then compare the resulting mesh to the original. We decided not to use this methodology, basically because the 3D scanner model is unrealistic, and the amount of effort required to get the original code working again, as well as interface properly with their program is significant.

Instead, the chosen methodology is to:

1. Start from a polygonal mesh representation, and sample a point cloud using the Poisson Disk Sampling as implemented in Meshlab [Cig+08] by Corsini et al. [CCS12].

2. Recompute the normals using the algorithm provided by Meshlab, in order to simulate a more realistic pipeline.

3. Execute any experiment-dependant preprocessing of the data.

4. Reconstruct the point cloud.

5. Compare the reconstruction to the original model by approximating the Hausdorff distance as computed by the Metro mesh comparison tool, by Cignoni et al. [CRS98], using the Meshlab implementation.

The output of the Metro tool is twofold: a mesh coloring that signals the areas of higher or lower error, and the minimum, maximum, mean, and root mean square error of the Hausdorff distance approximation. The former will be used for qualitative evaluation, while the latter will be used for quantitative evaluation.

However, it is important to point out that this methodology evaluates the entire processing pipeline, and not just the individual NCH fitting methods. Hence, if errors are incurred due to the isoextraction process rather than the fitting, it is something that we want to quantify, because such an improvement can be made separately from the methods developed in this thesis.

In order to tackle this problem, a similar evaluation method is proposed strictly for the NCH representation: comparison of the ρ^+ and ρ^- sets will be done by computing the absolute value of the difference and looking at the distribution; 3D plots can be made by coloring the point cloud too.

Indeed, we have the NNCH method which is what we want to approximate, and we have a few algorithms for approximating it, the SP and Corrected SB algorithms. Since the output of both methods should be of the same size -neither of them decimate any points-, we can directly compare $\rho_i^{+,\mathrm{NNCH}}$ and $\rho_i^{-,\mathrm{NNCH}}$ (the NNCH fit for ρ_i) against for example $\rho_i^{+,\mathrm{SP}}$ and $\rho_i^{-,\mathrm{SP}}$, the fit produced by SP. We do this by computing $|\rho_i^{+,\mathrm{NNCH}} - \rho_i^{+,\mathrm{SP}}|$ and

looking at the probability distribution; maximum, minimum, mean, variance, Root Mean Square (RMS), and quantiles will be computed throughout the entire point set. The key idea is to illustrate the behavior of the algorithm quickly: the maximum, minimum, mean, variance, and RMS are good measures, but very prone to outliers, which is why we also use quantiles[1]. The RMS definition is as usual:

$$\text{RMS} = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}}.$$

Where $\hat{y}_t$ is the estimation (i.e. SP's fit) and y_t is the ground truth (i.e. NNCH's fit). It is important to point all that all aggregate measures are computed using 64 bit floating point arithmetic, in order to get better accuracy, and that the implementation uses the Boost C++ Accumulators library in order to ensure correctness.

For all experiments run, the default parameters (named as they appear in the pseudo-code) will be used as shown in table 4.1. All point clouds and meshes are inside the unit cube centered at 0 (so the choice of initial radius is good), and no re-scaling or transform is used. All normal vectors are always pointing inwards when input to the program, although they may be pictured as pointing outwards to ease understanding and show differences more clearly. The rationale for choosing each parameter value is as follows:

- In practice, we have taken δ to be arbitrarily small and still managed to get convergence without a significant performance loss, but chose 1×10^{-5} since that is the same as used in [Pet18], which allowed us to more or less compare results against their implementation. 1×10^{-5} is also close to the minimum representable 32 bit floating point number, 1×10^{-6} can be represented, but 1×10^{-7} can not.

- $t_{\max}$ was taken to be 30 because of the same reason, but in practice no model we have tried thus far took longer than 10 iterations per point with δ as chosen.

- $y_{\max}$ was set to 30 because we have never seen any model do more than 2 iterations per point. Most points in fact do only 1.

- k was set to 10 because we did not notice any significant contribution to the convergence when setting it higher.

Although all algorithms presented in this thesis are amenable to parallelization and even to GPU implementations, only the NNCH implementation is made parallel by using

[1] For example, if we have failed estimation for a single point by a large margin, but all others have very small error, we will see all the aforementioned measures heavily penalized.

OpenMP, otherwise the fitting is too slow. The same could have been done for our SP and SB implementation (it literally only requires addition of one compiler pragma), but there is a conscious decision not to do this in order to get more accurate time measurements.

Further work in this area is encouraged, especially in translating these algorithms to the GPU. According to the work by Ma et al. [MBC12] and Jalba et al. [JKT13], SB could be made to run in the millisecond range for point sets with roughly 200k-300k points.

Parameter		Value
Radius Convergence Measurement	δ	1×10^{-5}
Maximum Number of Iterations	$t_{\max}$	30
Maximum Number of Refinement Iterations	$y_{\max}$	30
Initial Radius (constant)	r_{init}	200
Maximum Number of Points in Radius Search	k	10

Tab. 4.1: Default parameters used for the experiments.

Whenever a comparison to Poisson Reconstruction is shown, the variant is always Screened Poisson Reconstruction as presented by Kazhdan and Hoppe [KH13] and implemented in Meshlab; parameters are fixed to Meshlab's defaults. We found Poisson to always produce very smooth meshes under this configuration; so we tweaked slightly with the parameters to find out if there was any "sharper" configuration; we did not manage to find one, except when using much larger point cloud sizes. The size directly impacts our experimentation speed, because of NNCH's high complexity, so we decided to keep the default parameters, albeit this implies that the comparisons with Poisson may be slightly unfair.

4.3 Evaluation of Shrinking Planes

4.3.1 Ideal Conditions

For our first experiment, the behavior with noise-free and heavily sampled point clouds will be tested. This is basically a correctness check, as well as a way to understand how these algorithms compare in optimal operating conditions. For this, we will reconstruct models using the pipeline as described in section 4.3, working up from very simple models into more complex ones.

Cube

As a first example, we consider a unit cube, centered at the origin as shown in fig. 4.3. There are several important facts to point out about these pictures before we begin the analysis:

- In picture (d), the radii of the MAs get progressively larger as the center radii are located closer to the center of the figure.
- In (c), notice how the actual cube is entirely covered by green points. Those are all located at the border of the cube and have an infinite radius, concretely, they are actually responsible for the reconstruction later on, all other points could be removed.
- Both (e) and (g) show red when there is an approximation error between the reconstructed mesh and the original mesh from which the point cloud was sampled. Indeed, both pictures show that there are issues approximating the sharp edges in the cube. However, the Outer NCH has significantly less difficultly, and this is easy to see visually.

Perhaps the most immediate artifact that arises in these reconstructions is precisely how bad the border looks. There are several reasons why this happens. In the Inner NCH case (fig. 4.3 (d)), recall that we are approximating the surface as a union of balls contained within it; as mentioned in section 3.3.3, an infinite amount of points is required to approximate a sharp edge with balls.

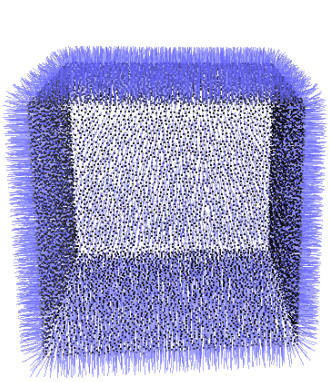
In the Outer NCH case (fig. 4.3 (f), figure (g) shows the Hausdorff error against the reference mesh as in (e)), the reason is more elusive: it is basically due to the fact that we are using MC, which is an algorithm well known to have issues with sharp edges; increasing the MC resolution will decrease the error and make it arbitrarily small, but will also take significantly more computation time. This issue will come up later on as well in many of the reconstructions, and it is the reason why many comparisons will be done directly between the NCHs: the resolution in the isoextraction algorithm accounts for a significant chunk of the error, while the NCH comparison speaks directly as to the approximation properties of our algorithm to the NNCH.

This kind of error can be fixed by using a more sophisticated isoextraction method (such as Dual Marching Cubes (DMC) by Nielson [Nie04] or Dual Contouring (DC) by Ju et al. [Ju+02]), a different visualization method (as done by Jalba et al. [JKT13]), or a

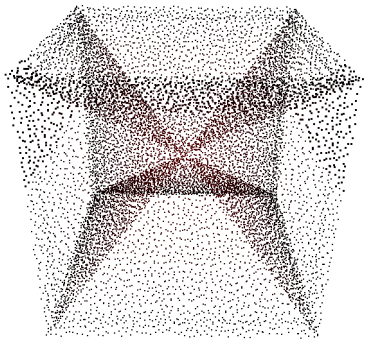
different polygonalization entirely. The implementation of a better isoextraction method is a complex endeavour, and thus was deemed out of scope for this thesis.

Finally, the perceptive reader will have noticed that no SB reconstructions are shown in the figure. This is because the results are too similar and do not produce any noticeable difference [2]; in fact, the approximate Hausdorff distance between the two meshes is 0 as per Metro. As a benchmark, in picture (e) the Hausdorff distance comparison between the NNCH reconstruction and the original mesh has a minimum error of 0, a maximum error of 0.010745, mean error of 436e-4, and RMS of 926e-4.

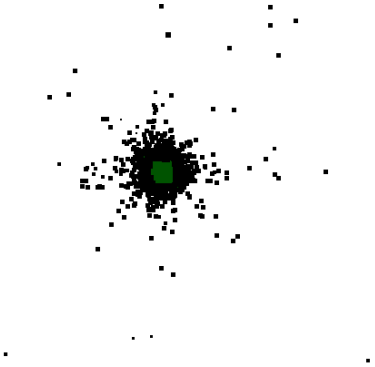
[2] From here on, it can be assumed that whenever such a result is omitted, it is because it adds no information.



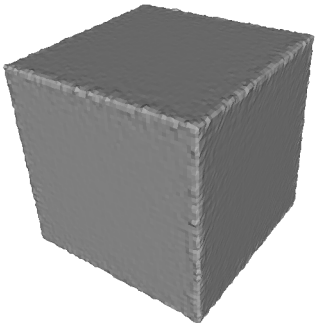
(a) Point cloud with 13k points. Normals depicted pointing outwards, reconstruction performed inwards.



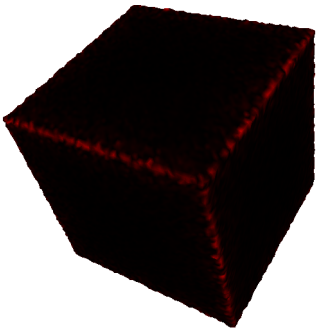
(b) Centers of its Inner Medial Atoms, as computed by NNCH. Red denotes ρ_i^+ after min-max normalization.



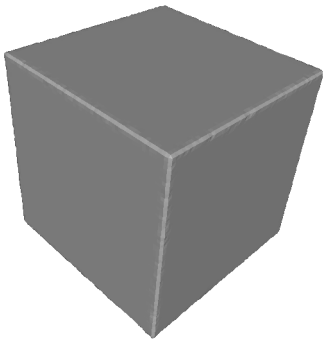
(c) Centers of its Outer Medial Atoms, as computed by SP. Green colored points are plane fits (i.e. $\rho_i^- = 0$)



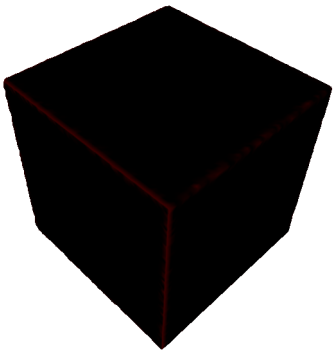
(d) Reconstruction of the Inner NCH, using NNCH.



(e) Hausdorff distance of Inner NNCH reconstruction vs original mesh.



(f) Reconstruction of the Outer NCH, using NNCH.



(g) Hausdorff distance of Outer NCH reconstruction vs original mesh.

Fig. 4.3: Cube reconstructed with NNCH and SP algorithms. Isoextraction performed by MC with low resolution (70) in all directions. The Hausdorff Distance between the NNCH and SB reconstructions is measured as 0 in Metro (the two reconstructions are equivalent), so the latter is not shown.

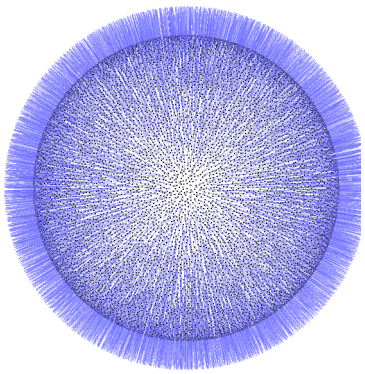
Sphere

As a second example, we consider a unit sphere, centered at the origin as depicted in fig. 4.4. There are three interesting things to remark here.

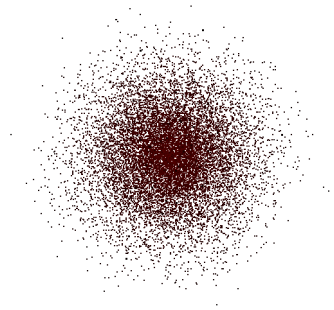
First, notice how the Inner MAs in (b) are not collapsed into one; this is a shortcoming of the fitting method. Ideally, we should have one single unit-sphere fit be output by the NCH fitting method, since the NCH is general enough to use just one sphere in order to cover all the points and still be correct. The fitting algorithm, however, generates one sphere per point, and many of them are different; furthermore, since all of them union into the same shape, most are not really contributing to the representation and could be omitted. From a theoretical perspective, this should never happen as it would be prevented by the maximality condition, however, in practice the sphere is represented as a polygon, the points are sampled with some amount of error, the normals are estimated from that (which is another cause of error), and we are also using floating-point arithmetic. This example is one strong argument for future work into an NCH-based simplification algorithm.

Second, notice how the Outer NCH in (c) is actually covering the border of the sphere with planes; indeed, the sphere is being reconstructed as an intersection of multiple linear half-spaces determined by planes, in a manner reminiscent of the Convex Hull.

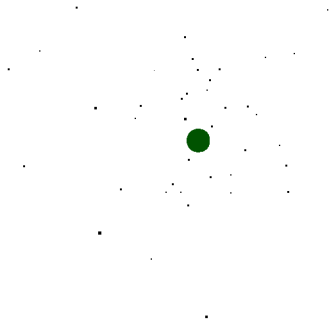
Third, there are circular error patterns in picture (d), which shows the Hausdorff Distance estimate for the point, where red means higher error. We have not managed to find a reason to explain these: they are unlikely to be due to the NCH reconstruction procedure itself (although they are produced by all methods tested), so the strongest hypothesis is that they are due to the Poisson Disk Sampling process, or Metro's facetting process, nevertheless the total error incurred is still very small.



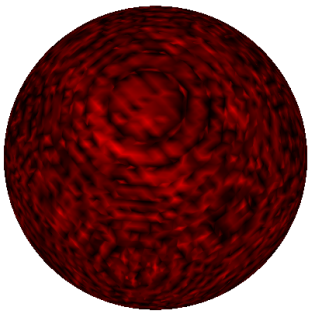
(a) Point cloud with 13k points. Normals depicted pointing outwards, reconstruction performed inwards.



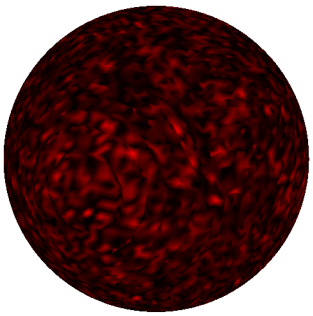
(b) Centers of its Inner Medial Atoms, as computed by SP.



(c) Centers of its Outer Medial Atoms, as computed by SP.



(d) Hausdorff distance of Inner SP reconstruction vs original mesh.



(e) Hausdorff distance of Outer SP reconstruction vs original mesh.

Fig. 4.4: Sphere reconstructed with the SP algorithm. Isoextraction performed by MC with low resolution (70) in all directions. The Hausdorff Distance between the NNCH and SB reconstructions is measured as 0 in Metro (the two reconstructions are equivalent), so the former is not shown.

Bunny

A slightly more complex example is that of the Stanford Bunny, shown in fig. 4.5. This example is more interesting than the previous ones, in particular because it is the first one with missing data (at the bottom of the model).

First, notice that the Outer MAs do not look like a fit to the shape, as it did in the previous examples. This is because there are not as many plane fits, which means that the points are drawn where the ball center should go. This renders the visualization basically useless in most cases, which is why we won't be using it in general, and will only show it when there is some insight.

Second, observe that there is a very significant quality change between a low resolution and medium resolution (as in MC's grid) reconstruction, which is very noticeable at the ears, in the intersection between the paws and the body, and the tail. This is very important to take into account when looking at any images from now on: regardless of the NCH fit, at the moment of rendering the model, the resolution used for reconstruction is one of the most important factors. This is the reason why we will measure the NCH differences instead of mesh differences in most cases, when measuring the NCH differences, we are looking into the approximation of SP of NNCH-computed values, instead of looking at two pieces of the pipeline at once.

	$\lvert\rho_i^{+,\mathrm{SB}} - \rho_i^{+,\mathrm{NNCH}}\rvert$	$\lvert\rho_i^{-,\mathrm{SB}} - \rho_i^{-,\mathrm{NNCH}}\rvert$
Maximum	0.971496582	0.92456454
Minimum	0	0
Average	0.00179697212	0.001036703
RMS	0.0244019292	0.0180983935

Tab. 4.2: Error Metrics for the NCH as computed by NNCH vs SB in the Stanford Bunny

	$\lvert\rho_i^{+,\mathrm{SP}} - \rho_i^{+,\mathrm{NNCH}}\rvert$	$\lvert\rho_i^{-,\mathrm{SP}} - \rho_i^{-,\mathrm{NNCH}}\rvert$
Maximum	0.00124931335	0.92456454
Minimum	0	0
Average	1.74973979e-06	0.000210608268
RMS	7.97802477e-06	0.010876026

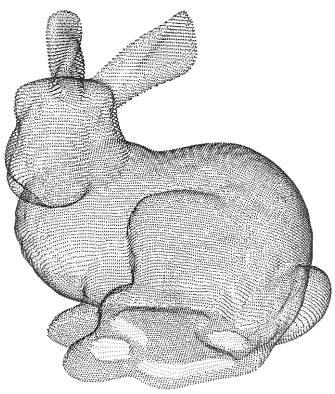
Tab. 4.3: Error Metrics for the NCH as computed by NNCH vs SP in the Stanford Bunny

In terms of the missing data, the results are more interesting than before, as shown in fig. 4.6. Notice the difference in fit on the under side where there is missing data, the

inner side subfigure (a) has protruding balls, while the outer (b) goes the other way, as expected. Also, notice how the symmetric NCH SDF makes a compromise between the two solutions.

As to the NCH fit itself, the approximation error of SP with respect to NNCH is shown in table 4.3, while the same table for SB is shown in table 4.2. Notice that the average error is small, which explains the equivalent reconstructions: it is more likely for errors to be introduced by the MC algorithm, than for it to come from the SP fitting.

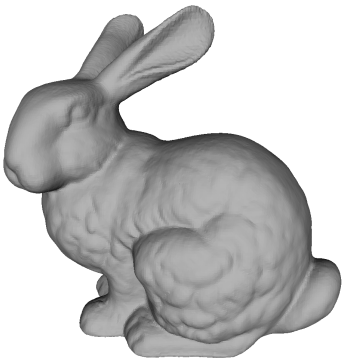
It is remarkable how the Outer NCH fit has significant higher error than the Inner NCH fit, but even under those conditions the average error of the SP algorithm is still an order of magnitude less than SB. Our hypothesis as to why this happens has to do with the fact that the Bunny in particular has high curvature in most of its body, thus making Inner spheres a much better fit than Outer spheres.



(a) Point cloud with 34k points and significant missing data under.

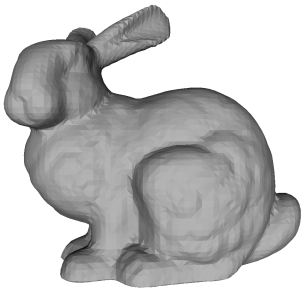
(b) Inner Medial Atoms, as computed by SP.

(c) Outer Medial Atoms, as computed by SP.

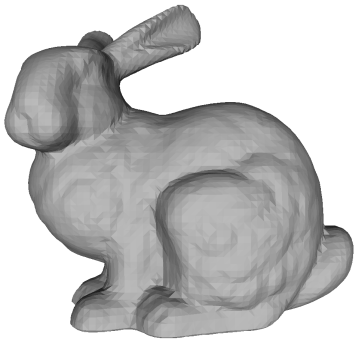


(d) Medium resolution (200) reconstruction of the Inner NCH computed by SP.

(e) Medium resolution (200) reconstruction of the Outer NCH computed by SP.

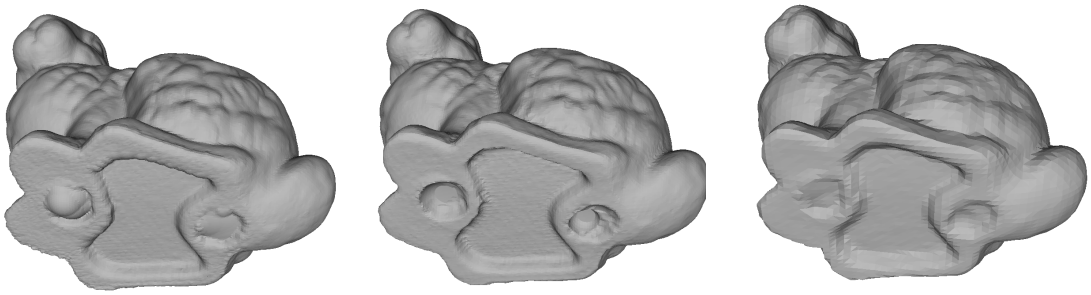


(f) Low resolution (70) reconstruction of the Inner NCH computed by SP.



(g) Low resolution (70) reconstruction of the Inner NCH computed by SP.

Fig. 4.5: Bunny reconstructed with the SP algorithm. The Hausdorff Distance between the NNCH and SB reconstructions is measured as 0 in Metro (the two reconstructions are equivalent), so the former is not shown.



(a) Underside of the Inner NCH reconstruction

(b) Underside of the Outer NCH reconstruction

(c) Underside of the Symmetric NCH reconstruction with an isoextraction resolution of 70

Fig. 4.6: Reconstruction by the SP algorithm.

Dragon

As a final example before the experiments, we display one fit that showcases the contributions of this thesis, which is that of the dragon shown in fig. 4.8. This figure is very complex, which in turn implies that it is very helpful for discovering bad behaviors and bugs in algorithms. As an additional feature, its 871k point cloud size implies that any slow operation will translate into a huge performance bottleneck within the tight inner loop.

The amount of points it has puts it severely out of the computing range of the NNCH algorithm, even when run multithreaded on a 4-core CPU. On the other hand, running the SB algorithm as originally published by Ma et al. [MBC12], Jalba et al. [JKT13], or Peters [Pet18] performs a bad Outer NCH reconstruction, as the output has several artifacts, shown in fig. 4.7. Aside from that, the proposed SP algorithm also fixes an artifact in the Inner NCH reconstruction, as made evident in fig. 4.8, pictures (c) and (d); this in particular is a direct benefit of the second refinement loop.

In terms of resource usage, obtaining the NCH for the dragon requires approximately 12 minutes of processing time per side (inner and outer) on both SP and SB, as measured in a Intel i7-6500U CPU with four cores at 2.50GHz. Memory usage remains constant at 220MB. Reconstruction is not performed with the Symmetric NCH because it is considerably slower than the Inner and Outer versions for such big models.

The processing time is way higher than state of the art methods such as Poisson Reconstruction, which takes 12 seconds for the same dataset; however, there are several differences here turning this into an unfair comparison:

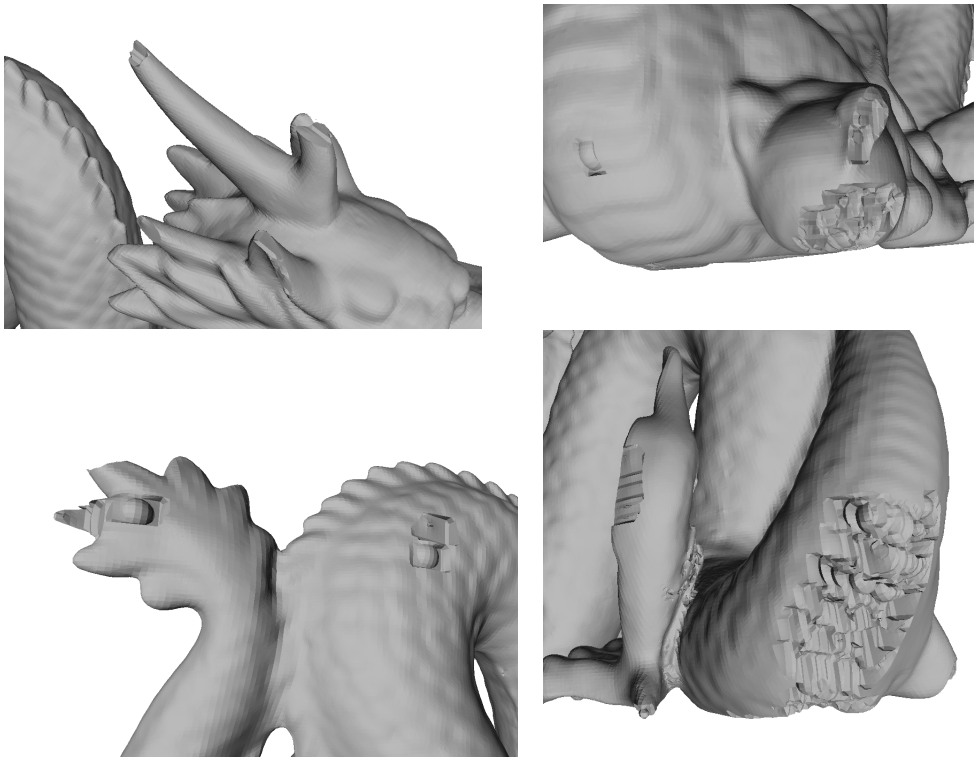
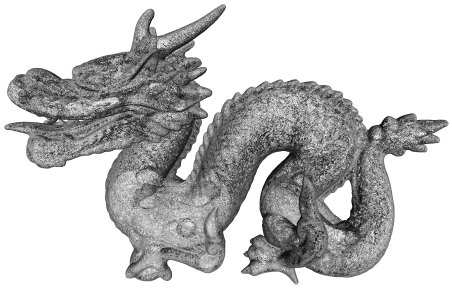


Fig. 4.7: Artifacts from the reconstruction of the Dragon's Outer NCH with the SB algorithm as originally published. Isoextraction performed by MC with a resolution of 300 in every direction.

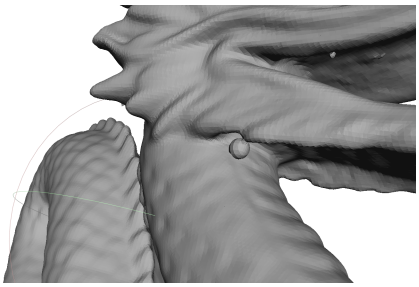
- The Poisson Reconstruction method is multithreaded and runs using multiple cores, which we are not doing.
- Poisson Reconstruction uses an Octree subdivision technique in order to perform adaptive reconstruction, this implies a much lower processing time due to heavy point cloud simplification happening in the process. For example, in the Dragon example (871k points), only 130k points are actually processed after decimation. We have not developed any such technique for this thesis due to lack of time, however, many ideas for this are discussed in section 5.1.
- The NCH is a much richer representation than what is produced by Poisson Reconstruction. When just using the MAs, it is possible to apply skeletonization techniques as in [JKT13], curvature motion [Lew+05] [Bor+09], and simplification [TH03], among many other applications.

Furthermore, the low memory usage of the SP algorithm, as well as its embarrassingly parallel approach, are the foundations for a extremely scalable reconstruction algorithm with provable guarantees. Lam [Lam16] already made a first step in this direction by analyzing out-of-core MAT approximation for LiDAR point clouds with the SB algorithm, but no such efforts have been made yet for surface reconstruction.

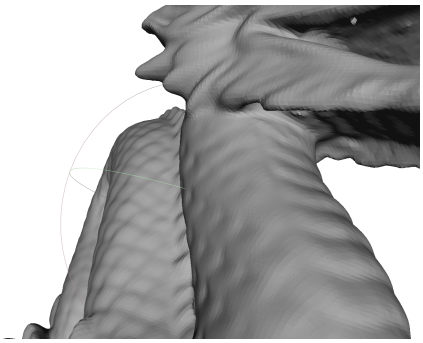


(a) The input point cloud, 871k points.

(b) Inner NCH by SP



(c) Reconstruction by the Corrected SB algorithm



(d) Reconstruction by the SP algorithm. Notice the removal of the spurious ball.

Fig. 4.8: Reconstruction of the Dragon's Inner NCH using MC with a resolution of 300 in every direction.

4.3.2 Noise

One of the most common problems when reconstructing real point clouds is that of the noise already present in the cloud due to acquisition and scanning. In this section, we measure the behavior of the algorithm when fitting point clouds that have increasing amounts of noise.

Before we begin our exposition, it is important to remark that since the NNCH algorithm is a global algorithm which fits maximal spheres or planes, and the SP and SB algorithms as presented here attempt to approximate it as closely as possible, they are all very likely to be susceptible to noise, basically because they over-fit: as shown in fig. 3.11, even a very small amount of noise can completely change the SB fit to a given point. Furthermore, as we mentioned earlier, the MAT is very sensitive to small deformations, so when fitting under noise, it is unlikely that the NCH (being similar to the MAT) resembles the NCH of the original object.

When adding noise, we will use Meshlab's Random Vertex Displacement functionality, which takes a Maximum Displacement parameter d, and adds to each point a random vector whose maximum norm is bounded by d. Concretely, it adds a uniformly sampled vector inside the closed ball $B_d(\mathbf{0})$ to each point in the cloud. As $d \to 0$, the ball collapses into 0, and thus the uniformly sampled point is small; when d grows, the ball becomes greater and thus the probability of choosing a point that is far from 0 grows along with d.

Cube

As a first experiment, we are interested in analyzing the behaviour of our algorithm in the simplest of settings, with varying amounts of noise. Thus, we chose the cube once again, as it is a easy shape to diagnose. We sampled 1k points from a unit cube centered at the origin, after which we added varying amounts of noise using Meshlab's Random Vertex Displacement, and recomputed the normals [3]. Reconstruction was performed for SB, SP, and NNCH, and then the resulting mesh was compared against the original mesh using the Hausdorff Distance approximation by Metro, with a 300k point sample. The results can be seen at fig. 4.11.

Notice that the first plot (a) appears to only show a single line (for the NNCH algorithm), however, this is not the case: the results were so similar that the plotting software showed only one line. Remember that this is a unit cube, so a 0.1 maximum displacement is roughly a 10% error with respect to the original, which is a very large error.

[3] This implies that there is noise simultaneously in both the coordinates and the normal vectors

In the second plot (b), we distinguish between the SDFs used to reconstruct the mesh. This plot shows that the Outer NCH is the reconstruction that achieves the least error as noise increases, with the symmetric variant being a close second. It is important to keep in mind that in this case, the surface is a cube, which is the best possible case for the Outer NCH, so it is expected that the reconstruction be better than the other two.

The fact that all algorithms achieve very similar error measurements means that the approximation performed by SP and SB is indeed very good in this case, and the NCH fit is as good as the original NNCH algorithm. Nonetheless, as shown in fig. 4.13, the reconstructions become very bad as noise increases, which validates our hypothesis. Notice in particular how the Inner NCH from no longer resembles the NCH of a well sampled, noiseless cube as shown in fig. 4.3. Moreover, it is worth emphasizing the huge difference between a 0.05 maximum displacement and a 0.1 maximum displacement (as seen in plots (a) and (b)); in particular, notice how the normals contain much more noise.

(a) Noiseless *(b)* 0.1 *(c)* 0.3

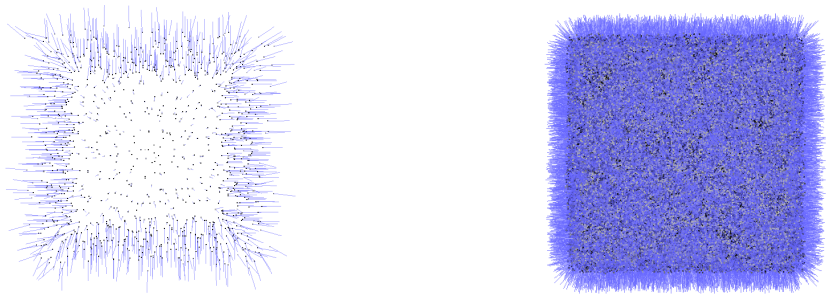
Fig. 4.9: Poisson reconstruction of a Cube as the Maximum Vertex Displacement increases

It is pretty clear that even a 0.05 maximum displacement is a lot for our algorithm to take in, even in a simple setting such as a cube. For comparison, the Poisson Reconstruction generates a rounded cube when the amount of noise is less than 0.1, and a sphere from 0.3 and onwards, as shown in fig. 4.9. Nevertheless, even though the cube is simple, it is also relatively pathological in comparison to smoother shapes, as we will see later.

As part of this experiment, we also performed the reconstruction with an increasing number of samples, in order to understand whether a denser sampling may help avoid the noise. The procedure for the generation was to perform Poisson Disk Sampling with an increasing amount of samples, add the Random Vertex Displacement, and then recompute the normals.

The results can be seen in fig. 4.12; Notice that low noise values actually benefit from a higher number of samples, while higher values may actually be disadvantaged when

adding samples. The reason this latter effect happens is because of the normal estimation procedure: this part of the pipeline is very sensitive to noise, and our methods are very sensitive to noise as well, and in particular to noise in the normals. This can be seen easily in fig. 4.10: the 1k sample has widely varying normals in comparison to the 130k. This convinced us to work with maximum displacement values up to and no more than 0.1. Notice that, as shown by the coloring of the bars in the plot, the Mean Hausdorff Distance does not increase significantly in the $[0, 1]$ displacement range, only the maximum does.



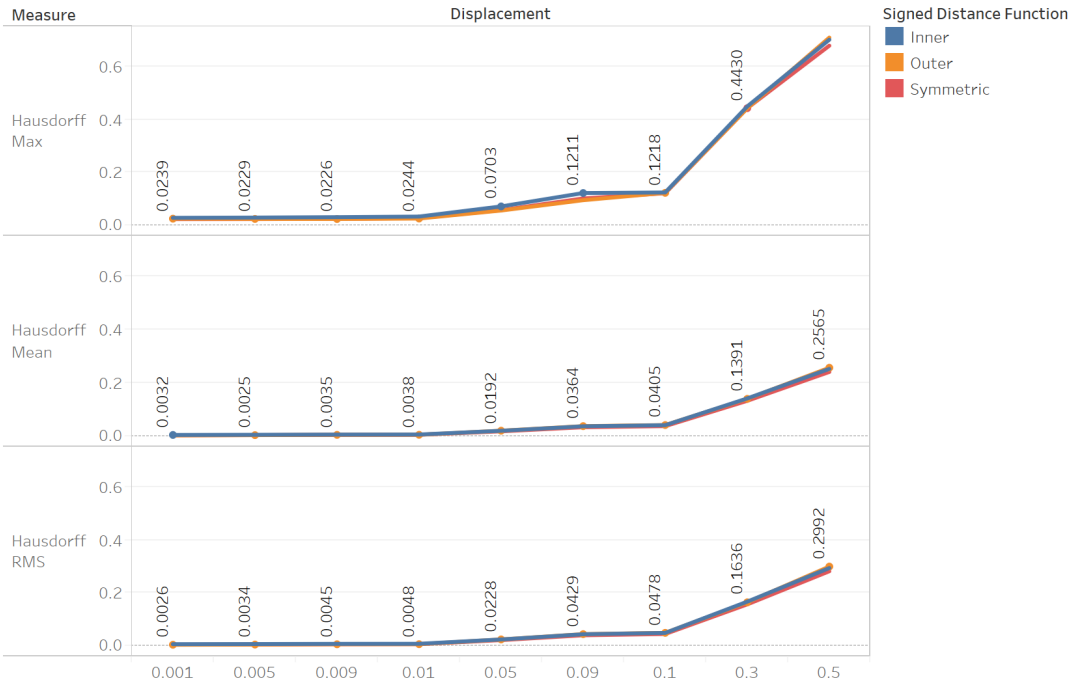
(a) A 1k samples Cube with 0.05 Maximum Vertex Displacement

(b) A 130k samples Cube with 0.05 Maximum Vertex Displacement

Fig. 4.10: A Cube point cloud after vertex displacement and normal estimation

We attempted to evaluate the corrected DSB in the same data set, but found that the algorithm's parameters were hard to select for a good reconstruction quality and would require further experimentation in order to succeed. This points out that noise resistance is a huge point to be worked on in the future.

(a) Hausdorff distance, shown categorized by method and SDF used to fit the surface.



(b) Hausdorff distance, shown categorized by SDF only. Since all methods produce very similar values, a single representative value is chosen for the Y axis.

Fig. 4.11: Hausdorff measurements from randomly displacing points in the cloud and performing reconstruction. The minimum is not reported because it was always 0, and the point cloud used is a Cube with 1k points. The X axis corresponds to the Maximum Displacement of Meshlab's Random Vertex Displacement.

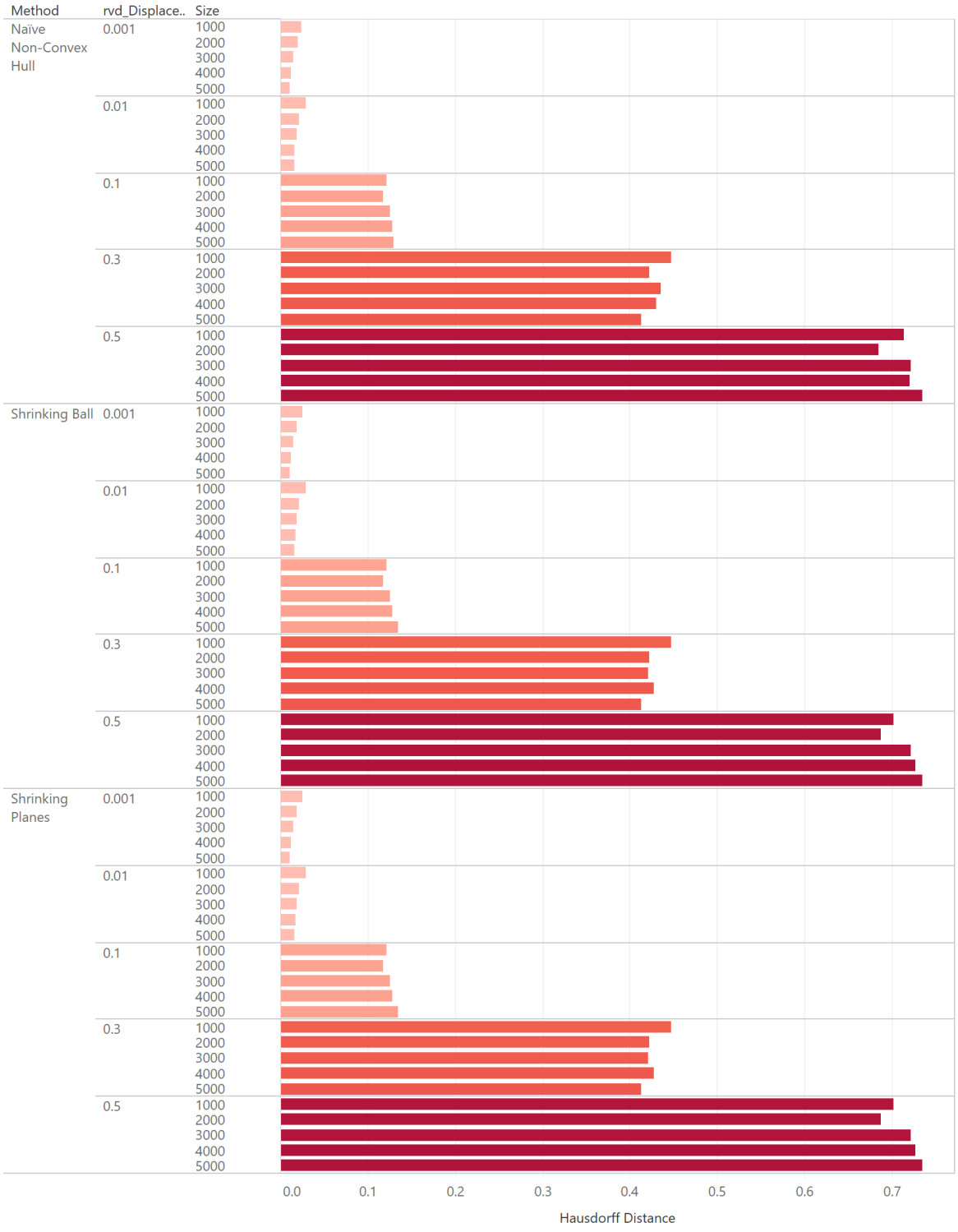
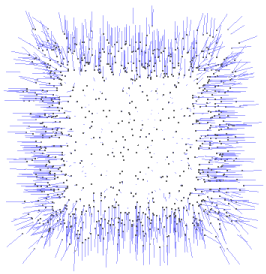
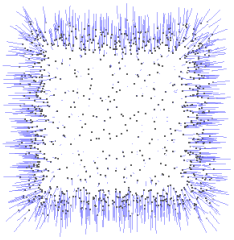


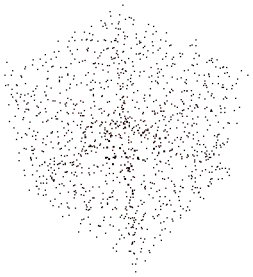
Fig. 4.12: The Maximum Hausdorff Distance, shown per method, displacement, and varying the size, while averaging the different SDF used. Color is assigned according to how high the average Hausdorff distance is.



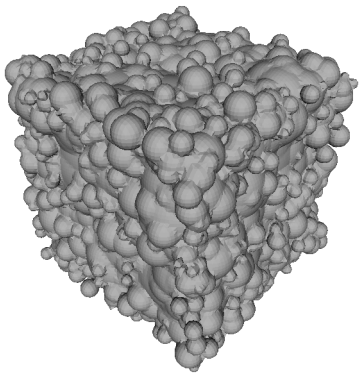
(a) Cube point cloud, 1k points with 0.1 maximum displacement.



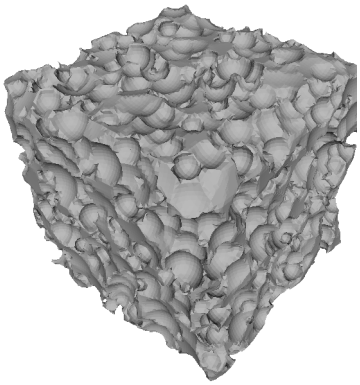
(b) Cube point cloud, 1k points with 0.05 maximum displacement.



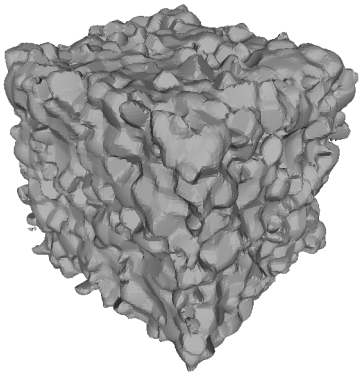
(c) Inner NCH as approximated by SP over the point cloud with 0.05 maximum displacement.



(d) Inner NCH with SP



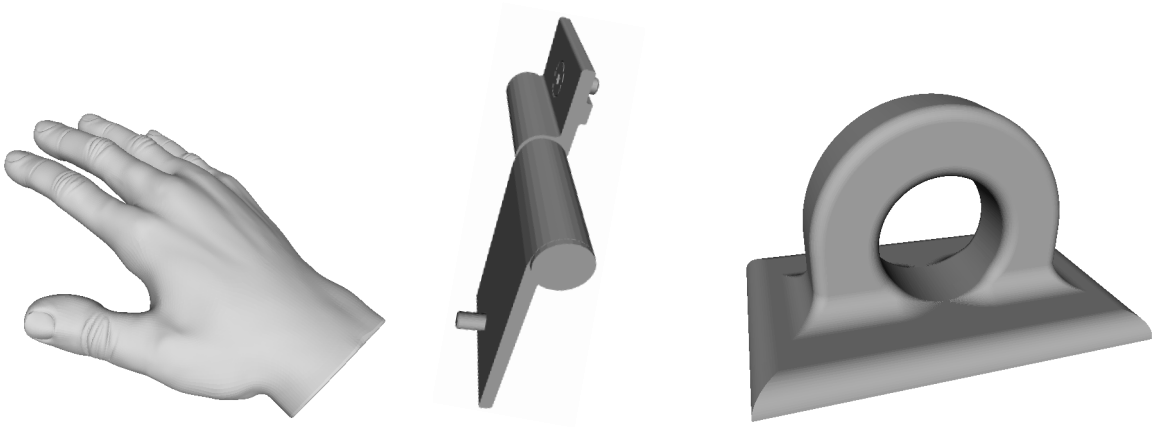
(e) Outer NCH with SP



(f) Symmetric NCH with SP

Fig. 4.13: Reconstruction of a Noisy Cube sample with 1k points and 0.05 maximum point displacement.

Multi-model Comparison



(a) Hand mesh used as our reference smooth model.

(b) Hinge mesh used as our reference model with sharp features. Shown rotated due to page size restriction.

(c) Key Eye Pad mesh used as our reference smooth model with large planar regions

Fig. 4.14: The reference models used for experimentation. In all cases, a 40k point cloud sample will be used throughout the experiments, since we found this size to visually preserve all features in the models.

For our next experiment, we will take three models, shown in fig. 4.14: one with smooth features but large planar regions, one with sharp regions, and one with just smooth features. We will sample point clouds for each one, and add different amounts of noise to each one, and perform reconstruction using SB, SP, NNCH and Screened Poisson; with an MC resolution of 50x50x50. After this, we will compare all results against the original source mesh using Metro, and also compare the NCH as fit by SP and SB against the reference NCH produced by NNCH. This will allow us to understand several things:

- The algorithms' behavior on different classes of shapes.
- SB and SP's NCH approximation capabilities (i.e. how good they are at obtaining the same NCH as NNCH).
- How the algorithms compare against a state of the art method such as Poisson from a quantitative standpoint.

We have several hypotheses that we hope to verify:

1. Since the noise resistance has been so bad in a simple example such as the Cube, there is no expectation that these examples, which are more complex, will do any better at all. We anticipate that in the best case, the reconstruction will resemble the original object, albeit with severe deformation of its border as it happened in the Cube example.

2. We think reconstruction with normals pointing outwards is likely to perform better than with them pointing inwards, especially for planar regions. The reason for this is that when normals point inwards, we may suffer from the same behavior as in the classical examples of small deformations generating MAT instability, as presented by [CCM97]. When the normals are pointing outwards, a sufficiently dense sampling should intuitively have a smoothing effect over the noise, as the reconstructed surface is the complement of the fit: the size of the Medial Atoms is going to be inherently larger in the Outer NCH, thus generating regions that are "almost planar", which is a defining characteristic of many real-world objects.

3. The final reconstruction should be much better for the Outer NCH in the models with sharp features and large planar regions, while the Inner NCH should have a higher success in the smooth model. This is a natural expectation, because the Outer NCH is actually able to fit planes, which are exactly what we need for planar regions and sharp features. Moreover, regions of high positive curvature should be better approximated by the Inner NCH, while regions of high negative curvature should be better in the Outer NCH.

4. We also expect the Symmetric NCH reconstruction to produce an intermediate result in terms of Hausdorff Distance. This comes from the fact that the Symmetric version is roughly an average between the two. It is intuitive that would be slightly more noise-resistant than the Inner NCH, but worse than the Outer NCH.

5. As to the NCH approximations, we expect that the SP will produce a much better result than SB, as it happened with the Bunny model. We are very interested in what could happen with the Outer NCH approximation, since in the Bunny model the Outer version had a significantly higher error.

Before we move on, we want to emphasize that Metro comparisons from here on will be done using the Meshlab RGB color palette, shown in fig. 4.15. The rationale behind this decision is that it boosts errors in real objects better than the red palette we had been using for the simple shapes.

Fig. 4.15: The Meshlab RGB color palette is the default palette for Quality Mapping in Meshlab, and will be used throughout our experimentation. Blue means the least possible error, while Red means the highest achievable. The exact values that determine which color will be used will be mentioned whenever used.

For reference, noise-free reconstructions using SP are shown in fig. 4.16. Indeed, these are the best we can possibly do, so this has to be kept in mind when looking at reconstructions. Even these noise-free reconstructions give us some leads as to whether the hypotheses that we posed are true or false:

- Notice in particular the remarkably better performance of the Outer NCH in both the Key Eye Pad and Hand models, as well as its incredibly bad reconstruction of the Hinge model, where it even generates separate components. It is important to remark that the Hinge is a CAD model, so it has a pin inside of it, hence causing the problems in the Outer NCH reconstruction.

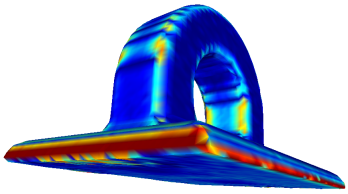
- The Symmetric NCH manages to find a compromise between the Inner NCH and Outer NCH in all three cases, at least visually.

- It may not be easily discernible in the images, but the Outer NCH is consistently doing a much better job at fitting planar regions than the Inner NCH, as expected. Likewise, the INCH is doing a better job at fitting the palm of the hand, and the cylinder of the hinge.

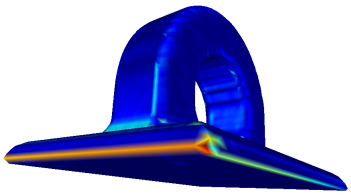
The results of this experiment are presented in several plots: fig. 4.18 for the Key Eye Pad, fig. 4.19 for the Hand, and fig. 4.20 for the Hinge. The NCH comparisons are shown in fig. 4.17

The first clear result from looking at the NCH approximation tables is that the algorithms are usually very good: notice that 99.999999% of the points have an error that is basically 0 for all practical purposes. Regardless, SP still wins over SB in basically every quantile comparison, with very few exceptions and not by a large amount.

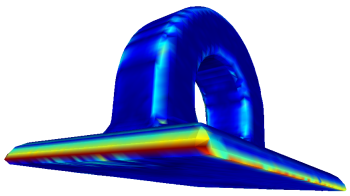
Observe however that the maximum is usually well above the quantile values, and this is because both methods fail at exactly one point. Nevertheless, SP is many orders of magnitude better than SB in most cases, which is to be expected as a direct result of the additional adjustment loop. Still, which algorithm has a better quantile error seems to be dependant on the specific shape.



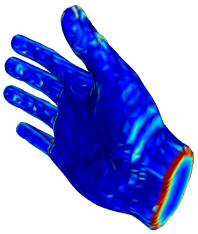
(a) Key Eye Pad's Inner NCH. Minimum value (blue) is 0, maximum value (red) is 0.003; other SDFs are drawn with the same scale.



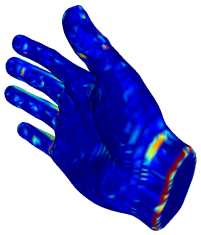
(b) Key Eye Pad's Outer NCH



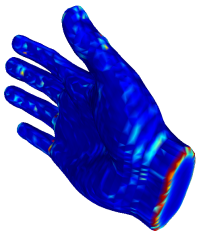
(c) Key Eye Pad's Symmetric NCH



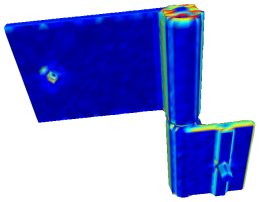
(d) Hand's Inner NCH. Minimum value (blue) is 0, maximum value (red) is 0.003; other SDFs are drawn with the same scale.



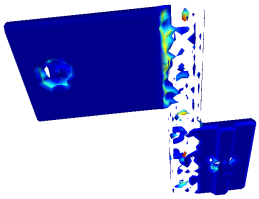
(e) Hand's Outer NCH



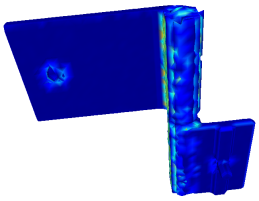
(f) Hand's Symmetric NCH



(g) Hinge's Inner NCH. Minimum value (blue) is 0, maximum value (red) is 0.008598.



(h) Hinge's Outer NCH. Minimum value (blue) is 0, maximum value (red) is 0.04342.



(i) Hinge's Symmetric NCH. Minimum value (blue) is 0, maximum value (red) is 0.031532.

Fig. 4.16: Noise-free reconstructions of point clouds sampled from the reference models. The process was performed using SP and MC with a resolution of 50 in all axes.

The Outer Maximum error is much larger than the Inner in all models. Indeed, this follows the same pattern as in the Bunny example. Moreover, SP is better than SB in this measure across the entire dataset, validating the fifth hypothesis. There is, however, one strange behavior happening in both the Key Eye Pad and the Hand models, which is that adding noise significantly reduces the Outer Maximum error, which is totally unexpected; on the other hand, the Hinge model does exactly the contrary.

Before analyzing the Hausdorff distance plots, we have to mention that the Inner NCH distance is missing when the Displacement is 0.1 due to a Metro bug that generates a Segmentation Fault in this case, for some reason.

Aside from that, notice that the 0 displacement cases establish our baseline: we can't be any better than that. Interestingly, notice that when there is no noise, there is at least one SDF that achieves either better or similar performance to Poisson.

Observe that the Outer NCH is usually on par with the Inner NCH or better, and the Symmetric NCH is always in the middle, which backs our hypotheses (2) and (4). Poisson Reconstruction is clearly superior in all scenarios, since even though the reconstruction error increases, it usually manages to keep the average error growing slower, which is a very desirable property.

A strange fact that we have noticed is that the Hausdorff Distance increases roughly linearly with the amount of noise introduced, and this seems consistent with the previous data as well. We have no reasonable explanation for this, as it is very counter intuitive that it should happen.

Model	Displacement	Method	Inner 99.999999% Quantile	Outer 99.999999% Quantile	Inner Max	Outer Max
Hand	0	Shrinking Ball	4.91481e-10	8.81201e-10	2.187201	2.849842
		Shrinking Planes	4.58247e-11	8.31102e-10	0.000120	2.849842
	0.025	Shrinking Ball	7.47301e-12	7.39889e-11	0.721527	0.551208
		Shrinking Planes	2.21067e-11	6.53276e-13	0.000244	0.003536
	0.05	Shrinking Ball	2.07961e-09	1.77800e-10	1.003372	1.141800
		Shrinking Planes	1.11757e-10	8.08200e-11	0.000366	0.005197
	0.1	Shrinking Ball	2.70708e-11	8.15042e-11	0.283401	0.436554
		Shrinking Planes	1.32013e-10	1.38656e-10	0.000244	0.000244
Hinge	0	Shrinking Ball	6.25722e-11	2.92809e-10	0.550804	0.455186
		Shrinking Planes	1.88220e-10	1.79519e-10	0.000046	0.455186
	0.025	Shrinking Ball	7.51624e-11	9.86684e-10	0.897903	2.346405
		Shrinking Planes	3.05402e-10	6.83187e-12	0.001465	0.007804
	0.05	Shrinking Ball	8.92577e-11	6.68842e-11	0.265167	0.894836
		Shrinking Planes	1.38533e-10	1.45203e-10	0.000488	0.004180
	0.1	Shrinking Ball	1.94900e-11	1.13571e-10	0.593491	0.975632
		Shrinking Planes	2.33398e-10	6.57283e-11	0.000366	0.005089
Key Eye Pad	0	Shrinking Ball	2.00655e-10	8.50938e-09	1.305176	2.831064
		Shrinking Planes	9.10384e-11	8.64880e-10	0.000154	2.831064
	0.025	Shrinking Ball	8.15398e-10	3.76865e-10	0.889664	0.602257
		Shrinking Planes	7.19459e-10	2.17372e-10	0.000366	0.012055
	0.05	Shrinking Ball	8.00246e-10	1.79451e-09	1.016327	0.791817
		Shrinking Planes	9.66616e-12	3.40917e-10	0.000366	0.006510
	0.1	Shrinking Ball	2.07915e-11	2.34796e-10	0.315674	0.408646
		Shrinking Planes	3.08548e-11	1.24428e-12	0.000244	0.004593

Fig. 4.17: NNCH's Rho approximation error per displacement and method for all models, as the maximum displacement increases. Color-coded using a Green-Red color palette per column, where Green coincides with 0, and Red with the maximum value. Although it doesn't make sense to compare numbers across different models, the colors are put in place to highlight cases in which the algorithms do particularly bad.

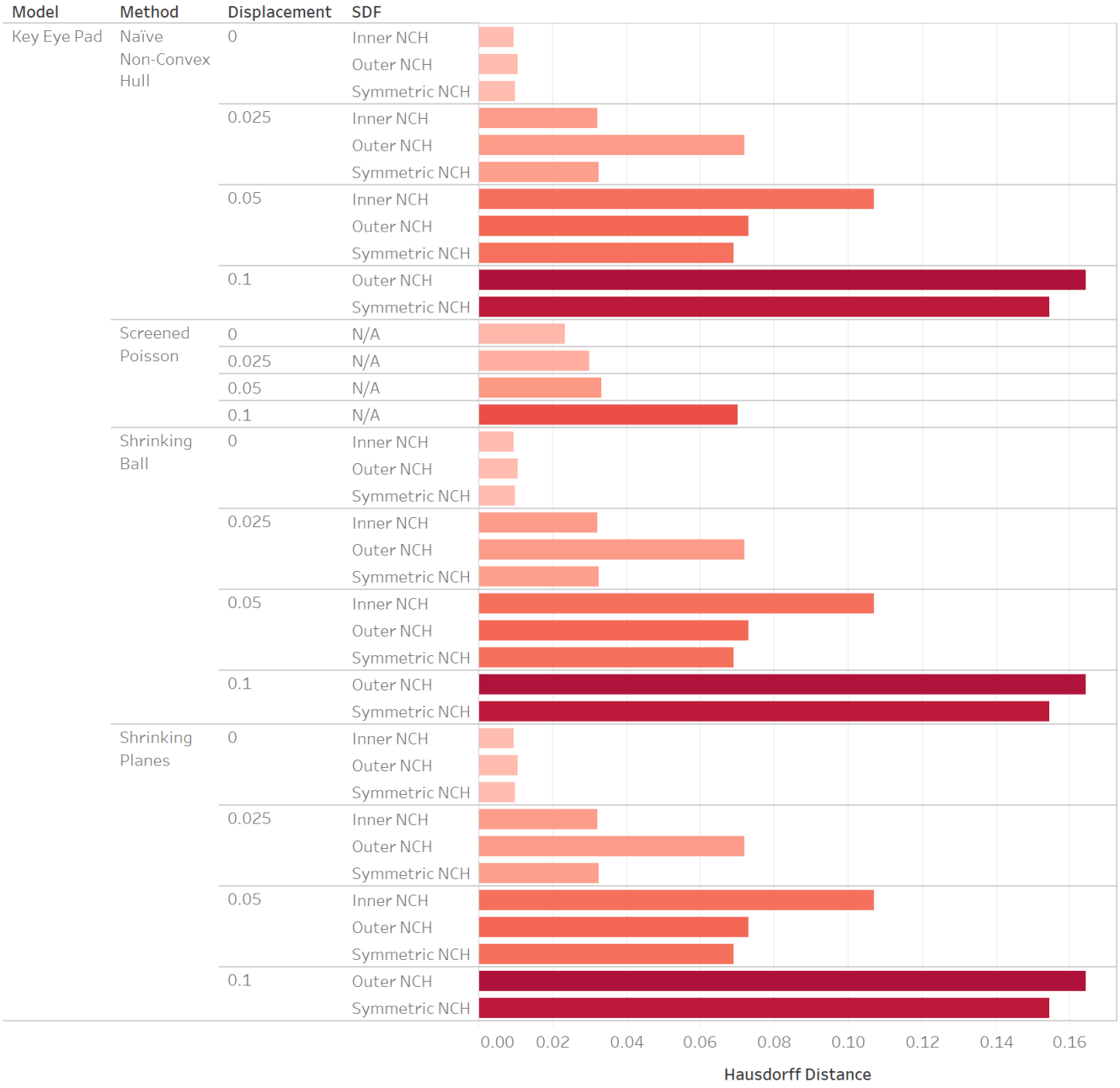


Fig. 4.18: Hausdorff Distance of the Key Eye Pad reconstruction against the reference mesh, for each method and SDF combination, as the maximum displacement increases. Color-coded using a White-Red color palette, where White coincides with 0, and Red with the maximum value, the color is assigned as per the Average Hausdorff Distance.

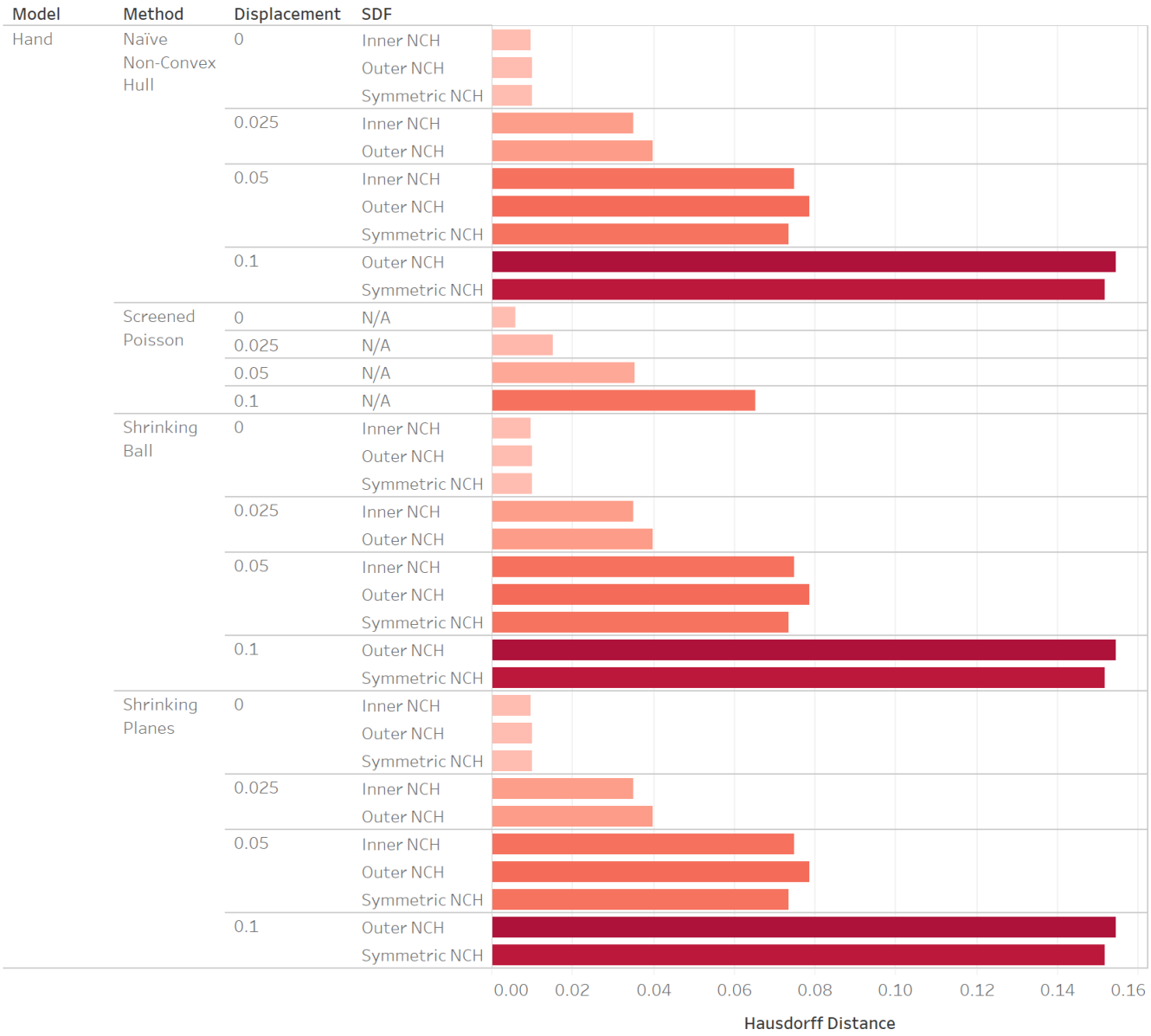


Fig. 4.19: Hausdorff Distance of the Hand reconstruction against the reference mesh, for each method and SDF combination, as the maximum displacement increases. Color-coded using a White-Red color palette, where White coincides with 0, and Red with the maximum value, the color is assigned as per the Average Hausdorff Distance.

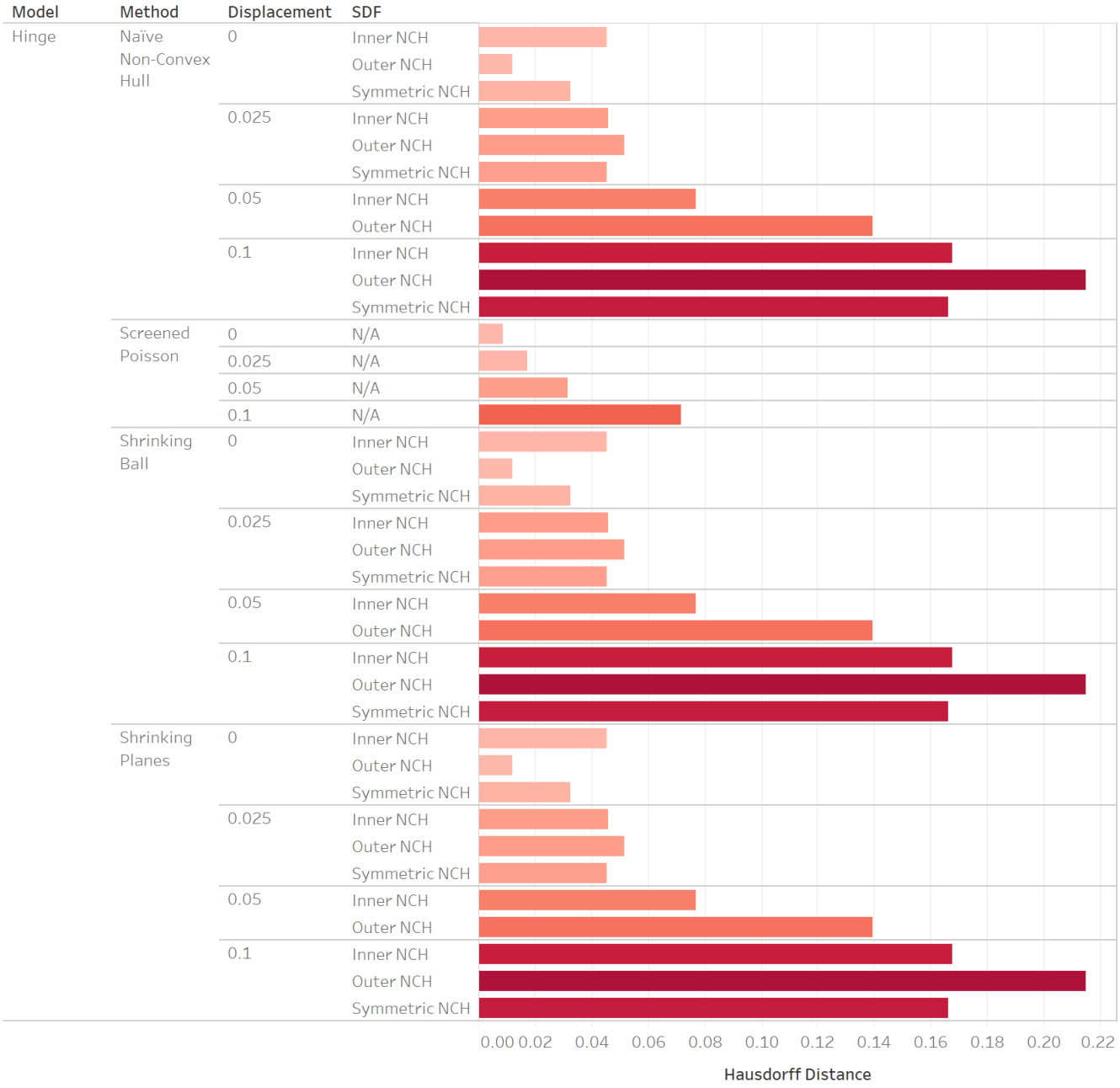


Fig. 4.20: Hausdorff Distance of the Hinge reconstruction against the reference mesh, for each method and SDF combination, as the maximum displacement increases. Color-coded using a White-Red color palette, where White coincides with 0, and Red with the maximum value, the color is assigned as per the Average Hausdorff Distance.

4.3.3 Missing Data

For our missing data experiments, we will manually remove points of a noiseless point cloud sampled from a given reference mesh, reconstruct the object, and then compare as we have been doing in the previous sections. The idea is to better understand what happens in practice as we provide less information to the algorithms.

The obvious hypotheses are that balls will be fit in the Inner NCH and will protrude from the surface, while the Outer NCH will instead have plane fits that will make the reconstruction look like a polygon. We also think it is likely for region removals to be suitably completed as long as no substantial error is introduced in the normal estimation procedure and removed regions are not exceedingly large.

We have experimented with the Cube, Sphere, Hand, Hinge, and Key Eye Pad models from the previous section, and performed several different removals for each in order to have good variety. The decimations made are as follows, with pointers to the results for each:

- Cube. Hausdorff computations for all cases are shown in fig. 4.24
  - Corner: we removed data from one of its corners fig. 4.22.
  - Side: we removed data in a rectangular shape, from one of its sides fig. 4.23.
- Sphere: we removed only a rectangular section of the border fig. 4.25. The Hausdorff computations can be seen in fig. 4.26.
- Hand. Hausdorff computations for all cases are shown in fig. 4.31
  - Palm 4.27: we removed a section from the palm of the hand. This is a smooth area which requires a change in orientation.
  - Pinky 4.28: the removal of the pinky shows how each algorithm completes a portion where there isn't really any data.
  - Wrist 4.29: all algorithms seemed to struggle with the reconstruction of the wrist border, even when points were present; it seemed an interesting case to check how each one completes this.
  - Wrist Cut 4.30: we expected this to have a similar behavior to that of the Cube's side, but wanted to see what would happen in a smooth surface.
- Hinge. Hausdorff computations for all cases are shown in fig. 4.35

  - Cap 4.32: the cap is similar to the Hand's pinky cut, but in a much sharper surface.
  - Cylinder 4.33: as mentioned earlier, the Hinge is a CAD model with a pin inside of it; through this decimation, the pin is exposed.
  - Disconnected 4.34: this removal splits the point cloud in two, showing how each reconstruction algorithm completes (or doesn't) the missing space.

- Key Eye Pad. Hausdorff computations for all cases are shown in fig. 4.40
  - Arc/Base Connection 4.36: the connection between the arc and the base has particular smoothness characteristics.
  - Base Corner 4.37: this corner case is different from the Cube's, since it is joins a smooth part and the planar base.
  - Inner Upper Arc 4.38: the decimated area is inside a very specific kind of geometric representation, so it is interesting to see how each NCH behaves in this case.
  - Upper Arc 4.39: this is the reflected version of the previous one.

It is important to remark that all Metro comparisons shown are colored using a scale where 0 is the minimum color and the maximum Hausdorff distance among all reconstructions is the maximum color. Moreover, the point clouds are shown from afar, so the black area highlights the removed part [4].

We also provide the NCH fit comparison in fig. 4.21. As in the previous experiments, we find that both SP and SB do a very good job approximating the NNCH, except for one outlier that sets the maximum. As in the previous cases as well, SP achieves a significantly lower maximum error in most Inner cases. Unlike the previous cases, the Outer Maximum for SP is always exactly the same as in SB.

It is important to remark that NNCH always finds the maximal ball given the available data, while SB and SP attempt to minimize as much as possible. Indeed, this means that the error should always be one sided: it must always happen that the ball found by SP and SB is bigger than the one found by NNCH. In particular, for the Outer NCH this means that the fit we find tends to make surfaces more planar, as the balls we fit are larger, and thus locally more similar to a plane. As to why there is such a significant difference

[4] The points of view of the point clouds differ from the Metro comparisons because the viewpoint of the former was chosen to highlight the decimation, while the latter to highlight the errors in all reconstructions

between NNCH and SP, it is due to the fact that the maximum number of points in the radius search has been constrained to 10, instead of set to ∞.

As to the reconstructions and the Hausdorff distances among shapes, it is worth noting that just as in the previous experiment, the reconstruction for the different shapes seems to be unaffected by the NNCH difference.

Of notable mention is that in almost all cases the methods presented in this thesis have managed to perform a better reconstruction than Poisson, at least in one of the variants. Moreover, as mentioned earlier the Symmetric NCH usually performs in between the Inner NCH and Outer NCH.

Notice that the Outer NCH handles very well the Cube cases, the Hand's Palm and both Wrist models, and all Key Eye Pad models. It retains its bad performance in the Hinge models, with the notable exception of the Disconnected Hinge version, in which it managed to fill in the disconnect better than the others, which was expected due to the nature of the method[5]. Aside from that, the Hand's Pinky in particular is fit with a clean cut, which is definitely not the best completion possible.

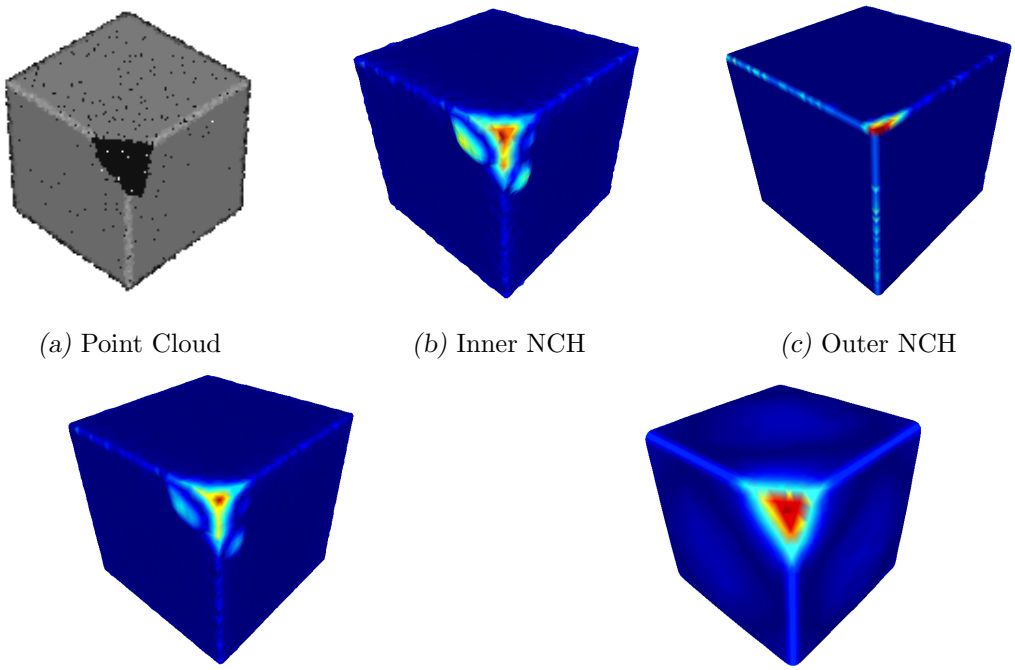
As to the Inner NCH, the fitting strategy transpires throughout the reconstructions. In literally every case with missing data, a wart is fit, as expected. This makes it a clearly bad choice in most cases, except in the Hinge model, where it performed a reasonable reconstruction for all but the Cylinder cut model.

The Symmetric NCH managed to produce intermediate shapes; in general, it seems like the Outer NCH is always a much better first choice. It is interesting that the SNCH actually managed to make a good reconstruction in the Hinge Cap model, comparable to that of Poisson. It is also certainly the best reconstruction in the Key Eye Pad Connection and Corners models.

[5] Since the normals are not forcing the shape to be closed, the ONCH is fitting planes for points at the border.

Surface	Model	Method	Inner 99.999999% Quantile	Outer 99.999999% Quantile	Inner Max	Outer Max
Cube	Corner	Shrinking Ball	1.13718e-10	4.35746e-07	0.002915	0.012844
		Shrinking Planes	5.29256e-11	4.35746e-07	0.000011	0.012844
	Original	Shrinking Ball	8.50309e-11	4.25505e-07	0.002915	0.008215
		Shrinking Planes	3.03243e-11	4.25505e-07	0.000011	0.008215
	Side	Shrinking Ball	2.14335e-11	4.09945e-07	0.002915	0.012844
		Shrinking Planes	4.85092e-11	4.09945e-07	0.000011	0.012844
Hand	Original	Shrinking Ball	1.93027e-10	8.24603e-10	2.187178	1.952737
		Shrinking Planes	3.45953e-11	1.20903e-11	0.000120	1.952737
	Palm	Shrinking Ball	1.88638e-10	1.57720e-09	2.586810	3.020227
		Shrinking Planes	2.11028e-11	2.19143e-11	0.000120	3.020227
	Pinky	Shrinking Ball	1.92162e-10	3.09685e-10	2.187178	2.706351
		Shrinking Planes	2.12766e-11	1.06463e-09	0.000120	2.706351
	Wrist	Shrinking Ball	3.56284e-10	5.74069e-11	2.187178	3.233138
		Shrinking Planes	5.58969e-12	1.13350e-09	0.000120	3.233138
	Wrist Cut	Shrinking Ball	1.90877e-10	9.17549e-10	2.187178	2.706351
		Shrinking Planes	4.95971e-11	1.51557e-09	0.000120	2.706351
Hinge	Cap	Shrinking Ball	5.39049e-11	1.98761e-10	0.550804	0.455186
		Shrinking Planes	1.35883e-10	2.76080e-10	0.000046	0.455186
	Cylinder	Shrinking Ball	2.00861e-10	5.32895e-11	0.550804	1.067172
		Shrinking Planes	1.32282e-10	5.61067e-11	0.000046	1.067172
	Disconnected	Shrinking Ball	1.92080e-11	1.75755e-11	0.550804	0.455186
		Shrinking Planes	2.76319e-11	1.43555e-10	0.000046	0.455186
	Original	Shrinking Ball	6.56673e-11	2.93504e-10	0.550804	0.455186
		Shrinking Planes	1.87599e-10	1.75527e-10	0.000046	0.455186
Key Eye Pad	Arc/Base Connection	Shrinking Ball	1.28581e-10	4.32424e-10	1.424606	1.697421
		Shrinking Planes	1.09496e-10	1.26722e-10	0.000046	1.697421
	Base Corner	Shrinking Ball	1.20554e-10	2.47534e-09	1.424606	1.697421
		Shrinking Planes	5.18026e-11	9.84727e-10	0.000046	1.697421
	Inner Upper Arc	Shrinking Ball	1.46417e-10	1.68964e-09	1.424606	1.362119
		Shrinking Planes	5.23301e-11	1.70363e-09	0.000046	1.362119
	Original	Shrinking Ball	2.47179e-10	5.61438e-10	1.424606	1.697421
		Shrinking Planes	1.07324e-10	5.59467e-10	0.000046	1.697421
	Upper Arc	Shrinking Ball	2.44904e-10	2.35384e-11	1.424606	1.362119
		Shrinking Planes	7.80262e-11	2.24667e-10	0.000046	1.362119
Sphere	Hole	Shrinking Ball	1.16832e-13	0.00000e+00	0.001145	0.000000
		Shrinking Planes	3.10739e-13	0.00000e+00	0.000105	0.000000
	Original	Shrinking Ball	1.11689e-11	0.00000e+00	0.001145	0.000000
		Shrinking Planes	1.89678e-13	0.00000e+00	0.000105	0.000000

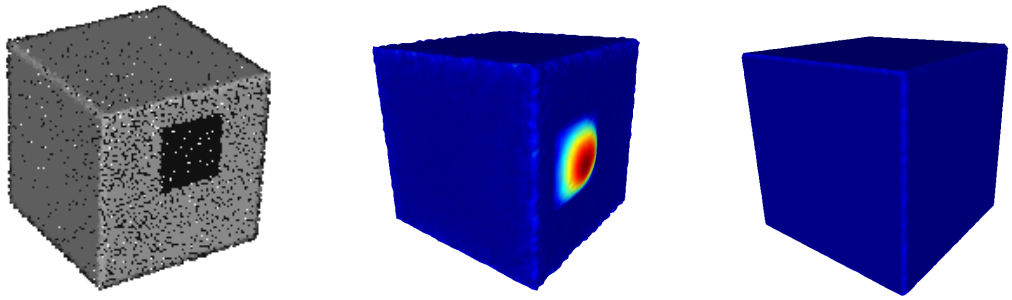
Fig. 4.21: NNCH's Rho approximation error per change for both SP and SP. Color-coded using a Green-Red color palette per column, where Green coincides with 0, and Red with the maximum value.



(a) Point Cloud *(b)* Inner NCH *(c)* Outer NCH

(d) Symmetric NCH *(e)* Screened Poisson

Fig. 4.22: Reconstruction of a Cube with missing data in its corner. The sample size is 10k, and the reconstruction was performed using SP and MC with a resolution of 50 in all axes. Each plot has an independent color scale because the differences are too large.



(a) Point Cloud *(b)* Inner NCH *(c)* Outer NCH

(d) Symmetric NCH *(e)* Screened Poisson

Fig. 4.23: Reconstruction of a Cube with missing data on its side. The sample size is 10k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions. The minimum value (blue) is 0, and the maximum value (red) is 0.05504, with the same scale used for all plots.

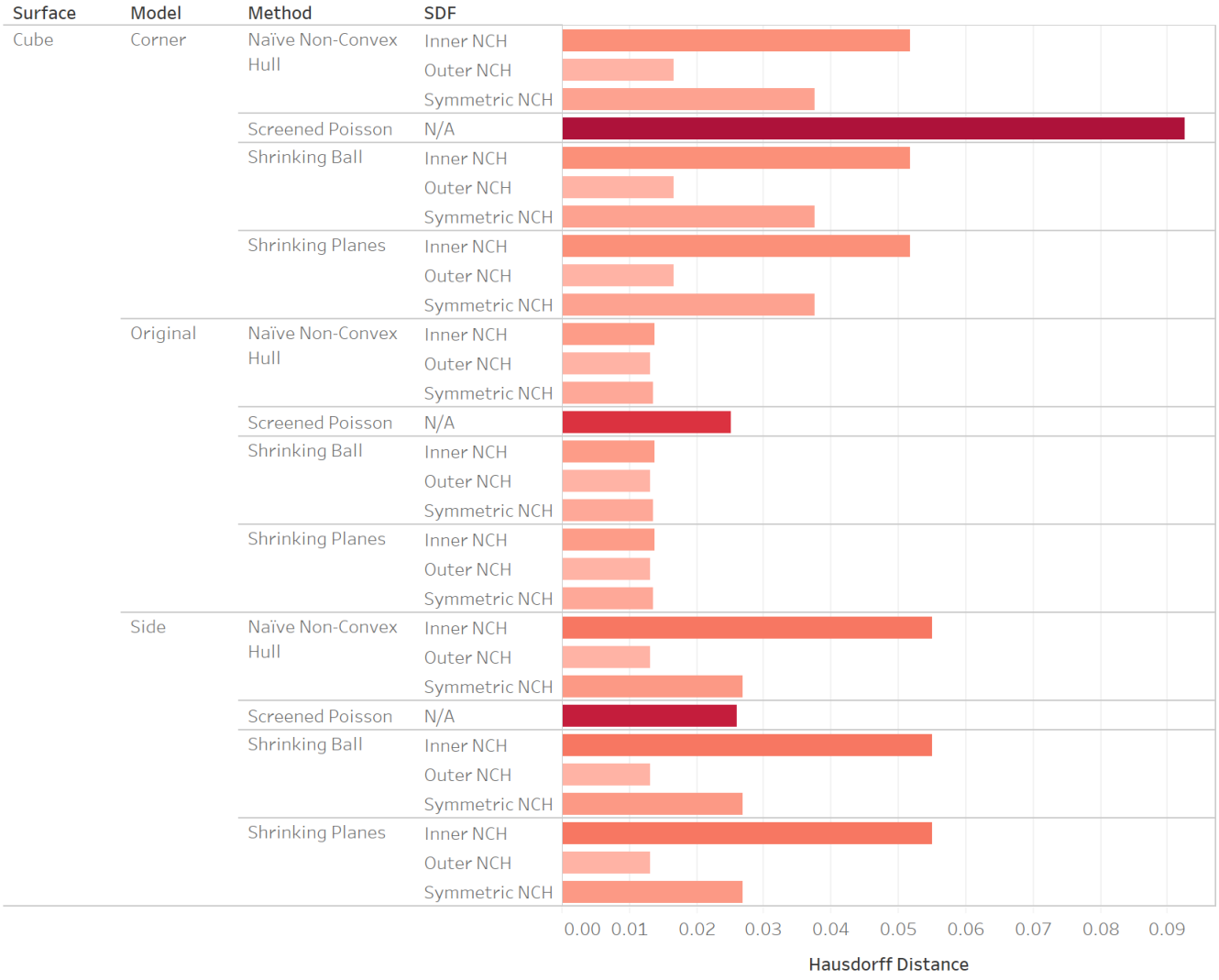
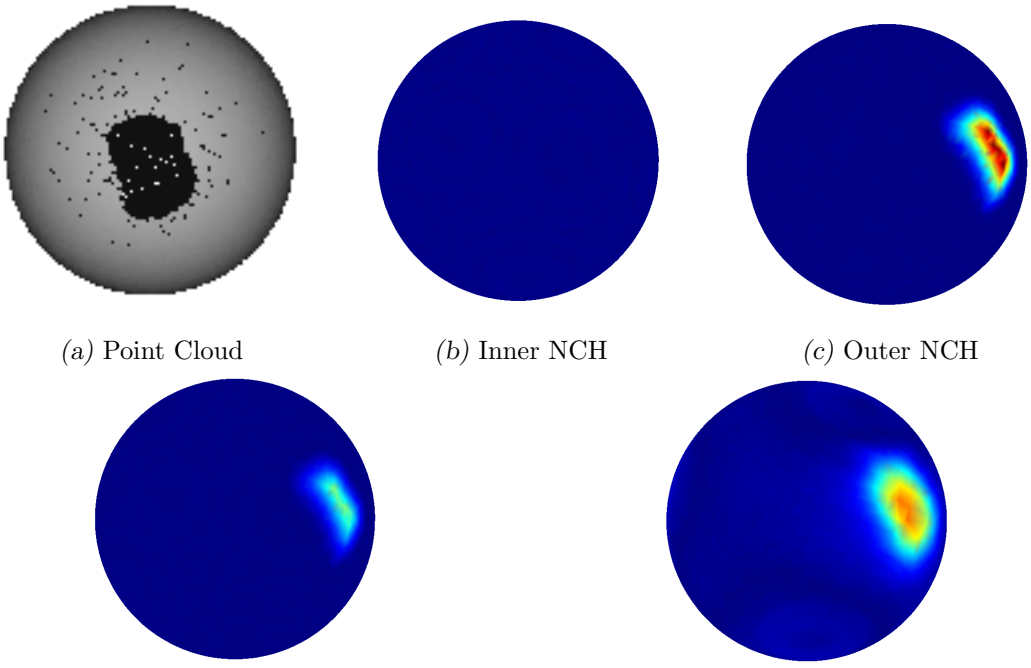


Fig. 4.24: Hausdorff Distance of the Cube reconstruction against the reference mesh, for each change produced. Color-coded using a White-Red color palette, where White coincides with 0, and Red with the maximum value, the color is assigned as per the Average Hausdorff Distance.



(a) Point Cloud *(b)* Inner NCH *(c)* Outer NCH

(d) Symmetric NCH *(e)* Poisson

Fig. 4.25: Reconstruction of a Sphere with missing data. The sample size is 10k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions.

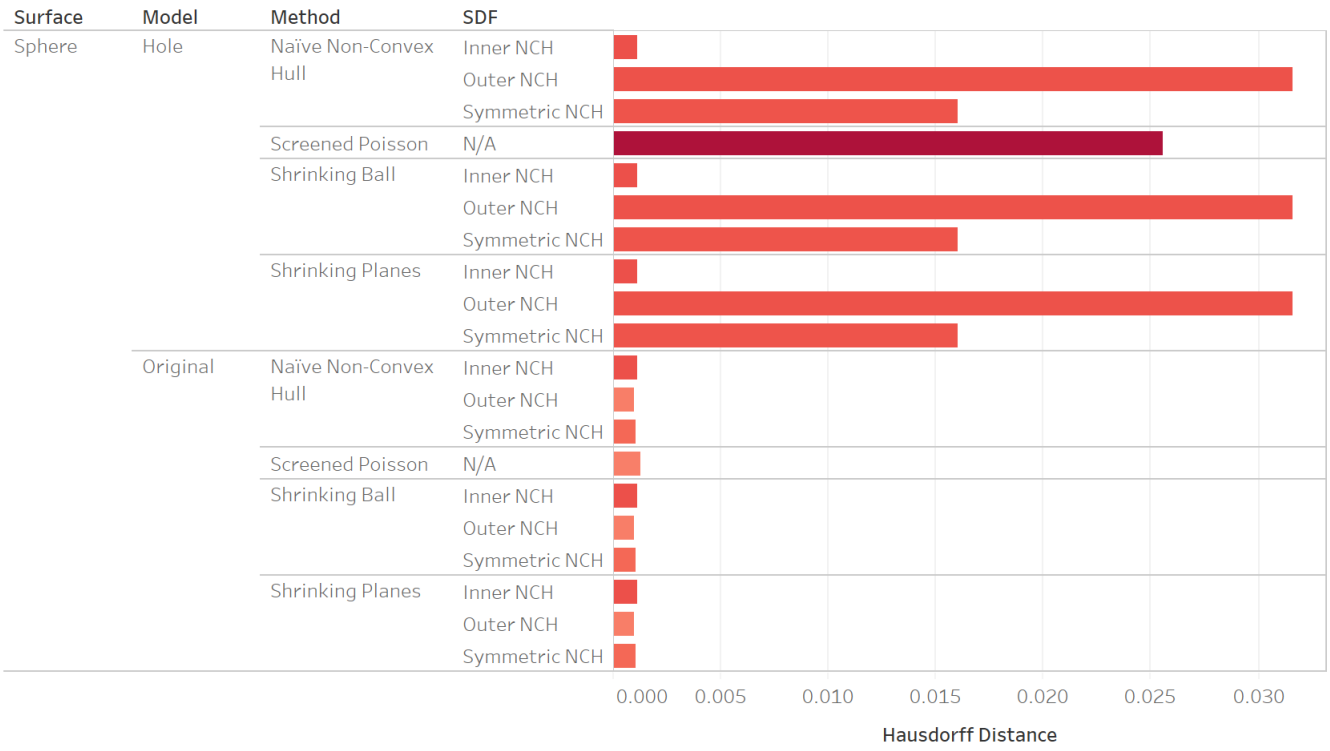
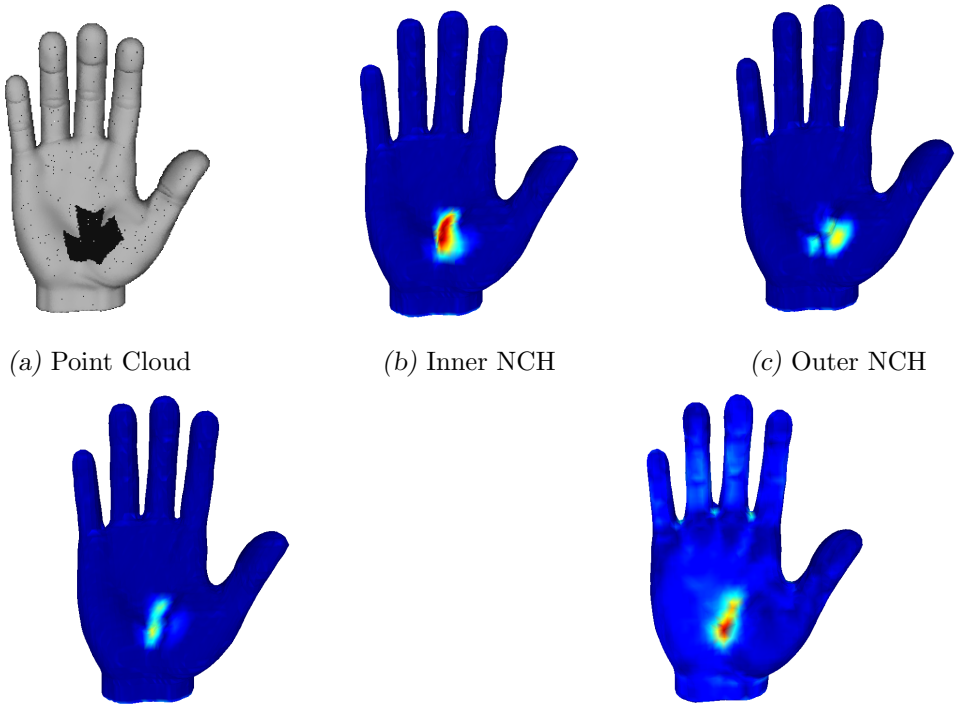


Fig. 4.26: Hausdorff Distance of the Sphere reconstruction against the reference mesh, for each change produced. Color-coded using a White-Red color palette, where White coincides with 0, and Red with the maximum value, the color is assigned as per the Average Hausdorff Distance.



(a) Point Cloud *(b)* Inner NCH *(c)* Outer NCH

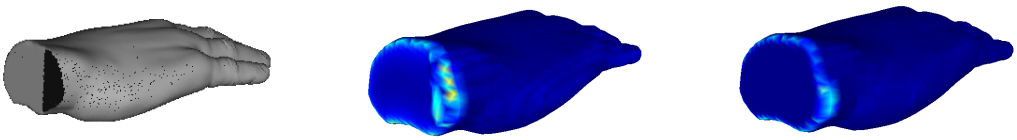
(d) Symmetric NCH *(e)* Poisson

Fig. 4.27: Reconstruction of the palm of the Hand with missing data. The sample size is 40k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions.

(a) Point Cloud *(b)* Inner NCH *(c)* Outer NCH

(d) Symmetric NCH *(e)* Poisson

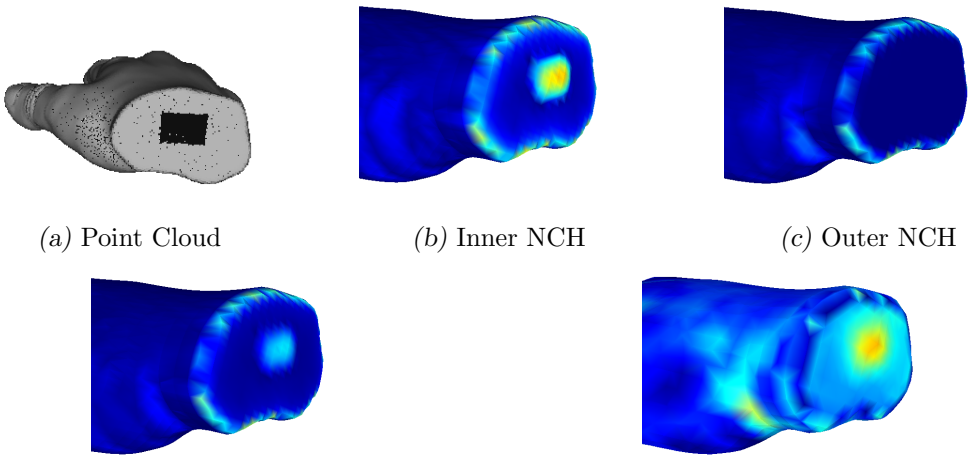
Fig. 4.28: Reconstruction of a Hand with the Pinky removed. The sample size is 40k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions.



(a) Point Cloud *(b)* Inner NCH *(c)* Outer NCH

(d) Symmetric NCH *(e)* Poisson

Fig. 4.29: Reconstruction of a Hand with the part of its wrist removed. The sample size is 40k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions.



(a) Point Cloud *(b)* Inner NCH *(c)* Outer NCH

(d) Symmetric NCH *(e)* Poisson

Fig. 4.30: Reconstruction of a Hand with a hole on the planar side of the wrist. The sample size is 40k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions.

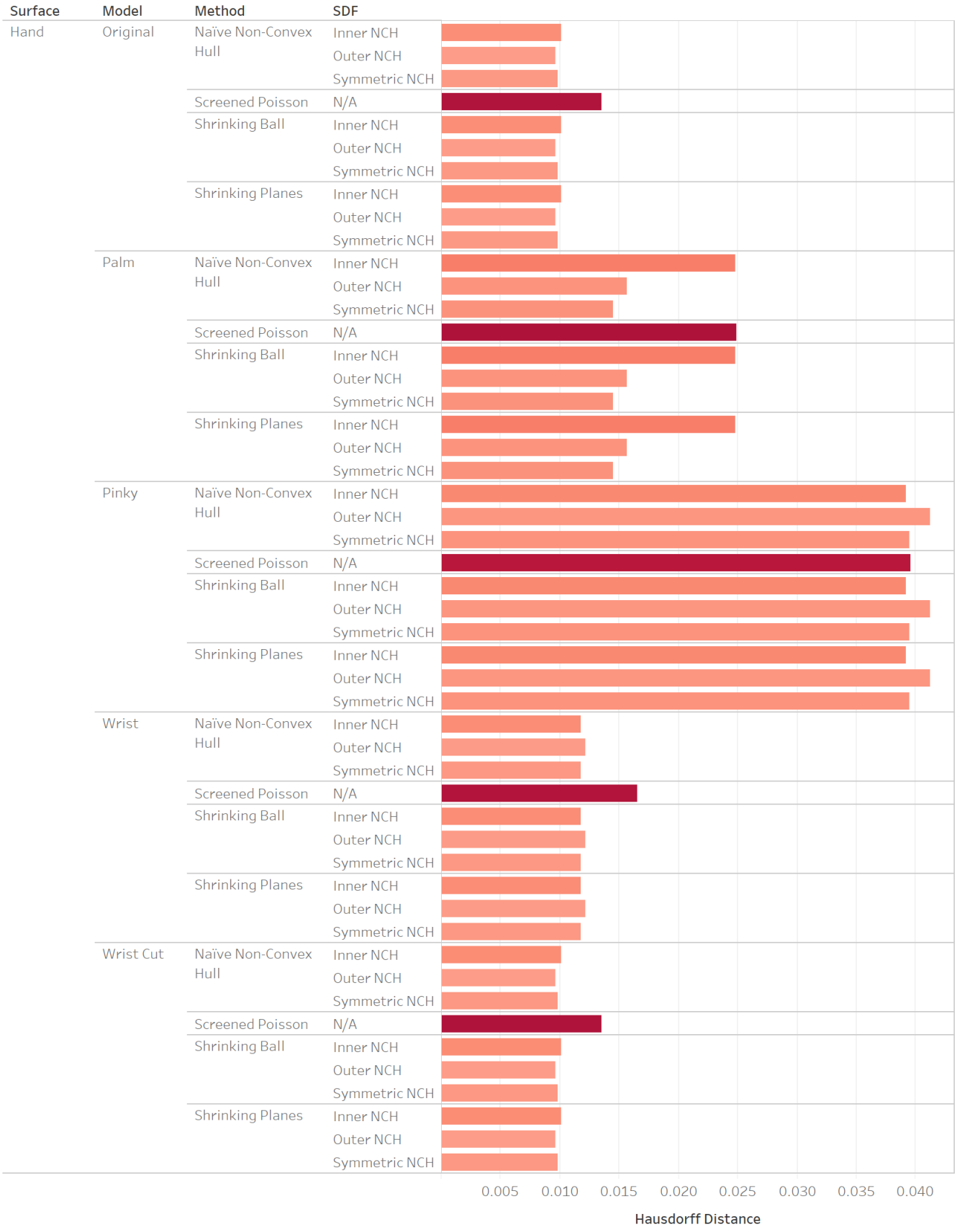
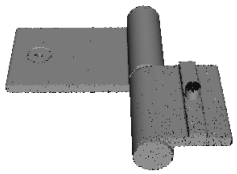
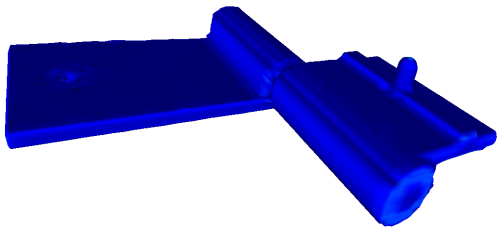


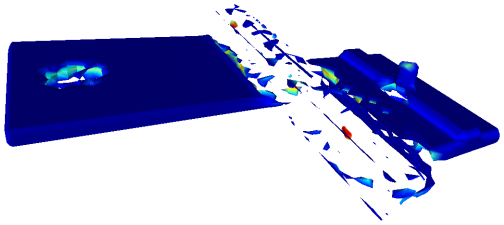
Fig. 4.31: Hausdorff Distance of the Hand reconstruction against the reference mesh, for each change produced. Color-coded using a White-Red color palette, where White coincides with 0, and Red with the maximum value, the color is assigned as per the Average Hausdorff Distance.



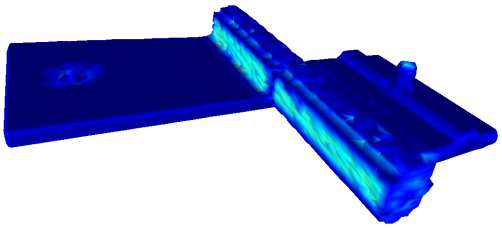
(a) Point Cloud



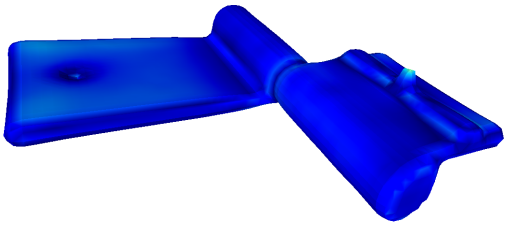
(b) Inner NCH



(c) Outer NCH

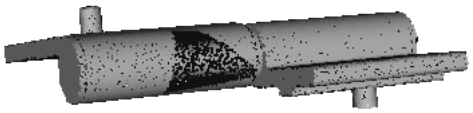


(d) Symmetric NCH

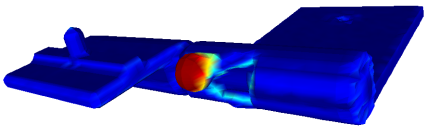


(e) Poisson

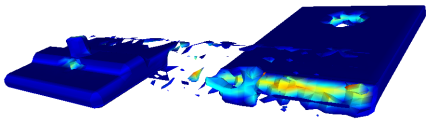
Fig. 4.32: Reconstruction of a Hinge with one of the protruberances' partially removed. The sample size is 40k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions.



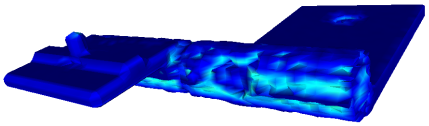
(a) Point Cloud



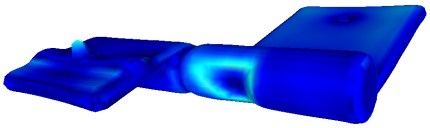
(b) Inner NCH



(c) Outer NCH

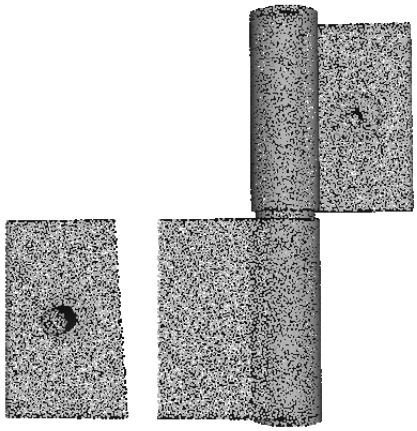


(d) Symmetric NCH

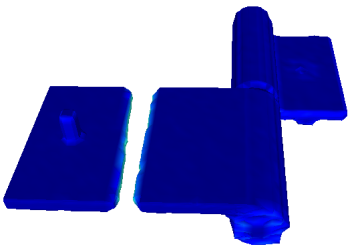


(e) Poisson

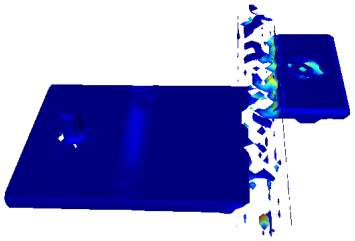
Fig. 4.33: Reconstruction of a Hinge with part of the cylinder removed. The sample size is 40k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions.



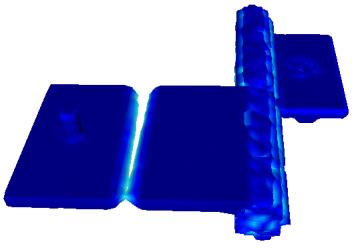
(a) Point Cloud



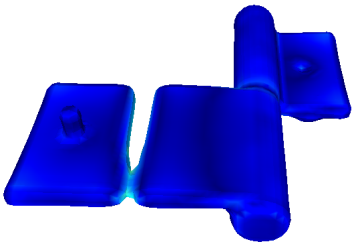
(b) Inner NCH



(c) Outer NCH



(d) Symmetric NCH



(e) Poisson

Fig. 4.34: Reconstruction of a Hinge with complete removal of a part of its sheet. The sample size is 40k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions.

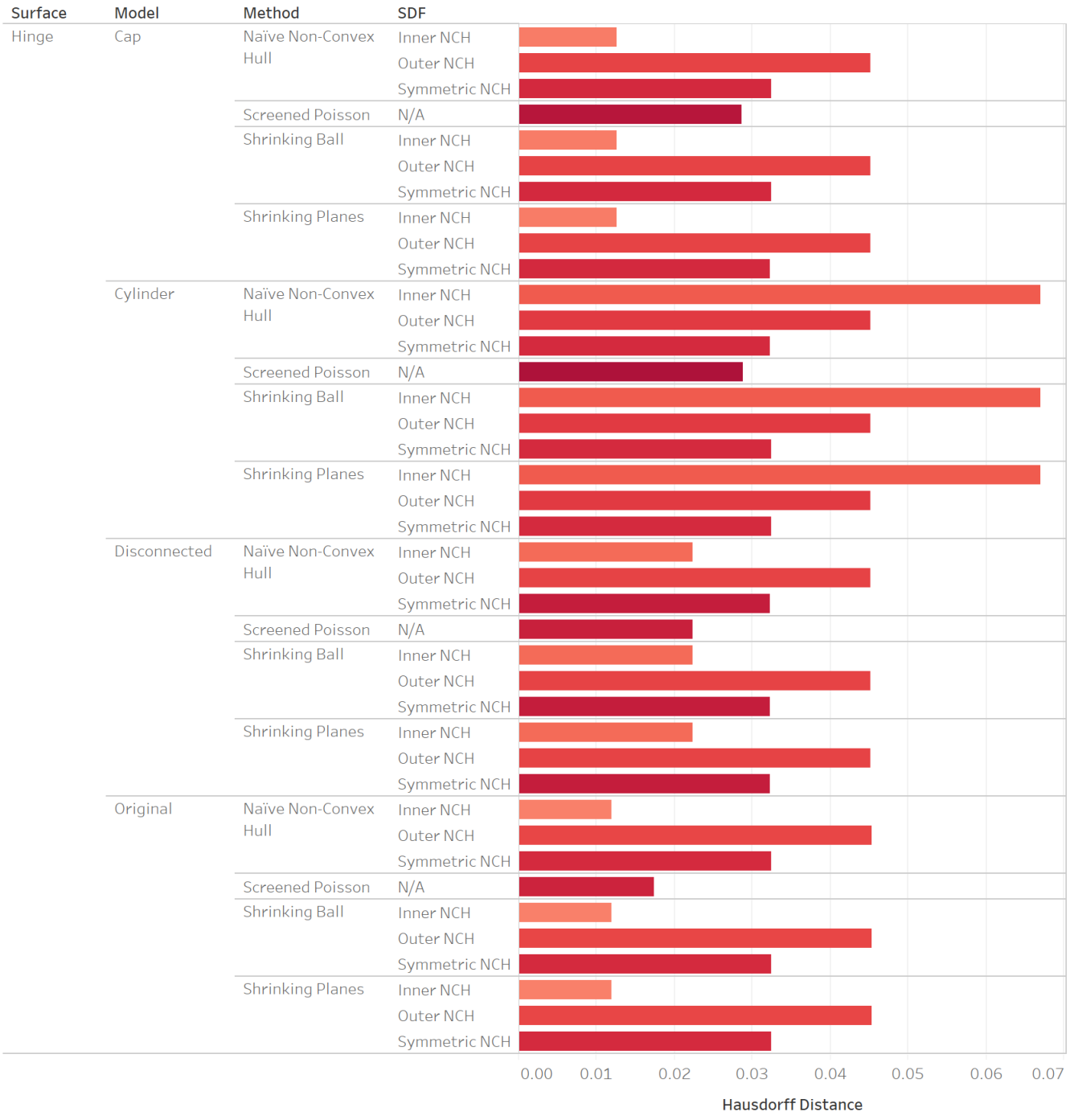
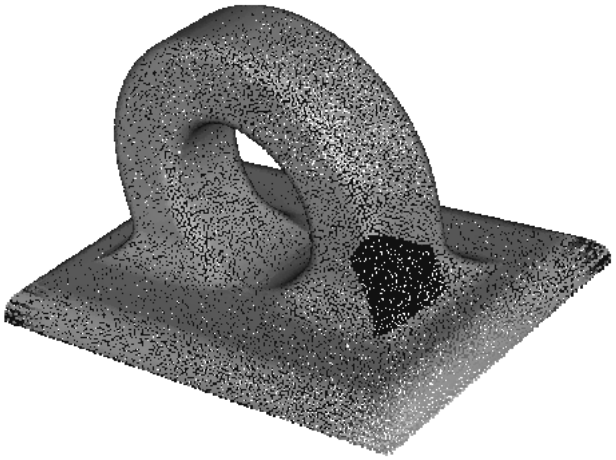
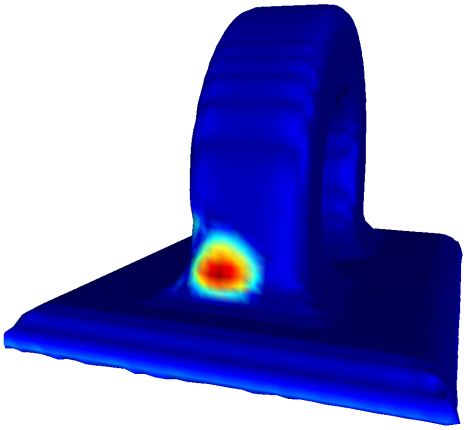


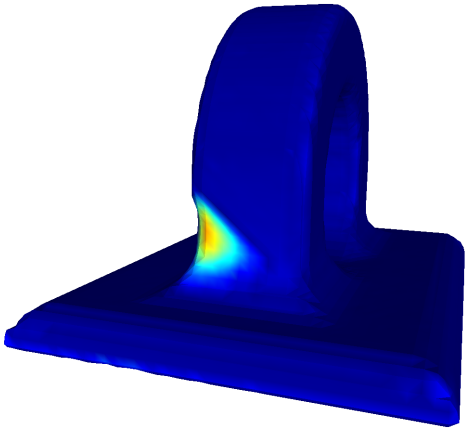
Fig. 4.35: Hausdorff Distance of the Hinge reconstruction against the reference mesh, for each change produced. Color-coded using a White-Red color palette, where White coincides with 0, and Red with the maximum value, the color is assigned as per the Average Hausdorff Distance.



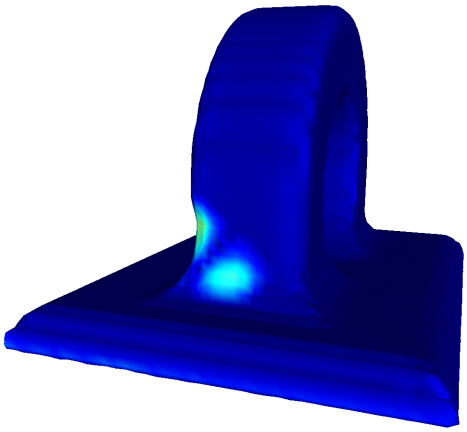
(a) Point Cloud



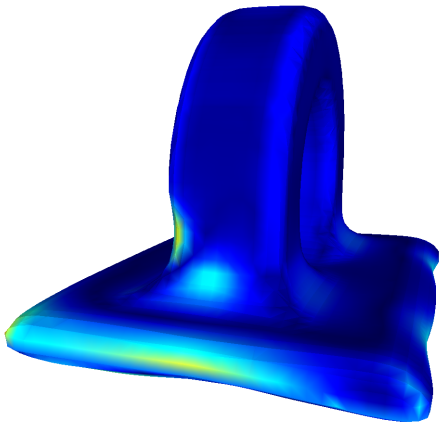
(b) Inner NCH



(c) Outer NCH

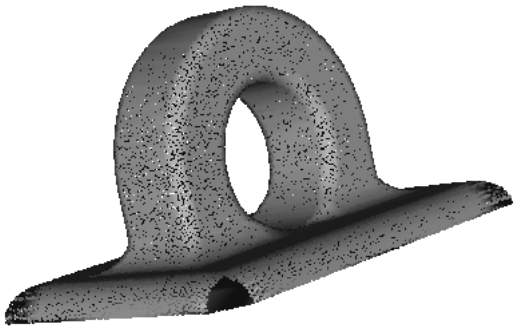


(d) Symmetric NCH

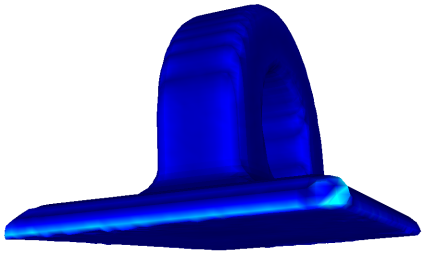


(e) Poisson

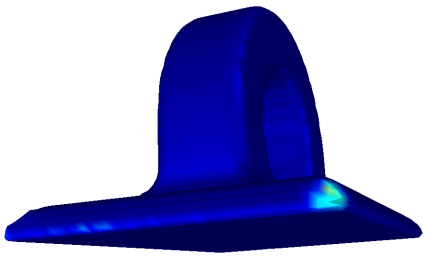
Fig. 4.36: Reconstruction of the Key Eye Pad with part of the connection between the arc and the base removed. The sample size is 40k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions.



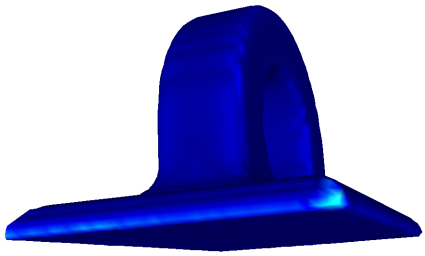
(a) Point Cloud



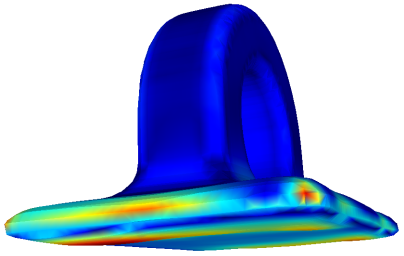
(b) Inner NCH



(c) Outer NCH

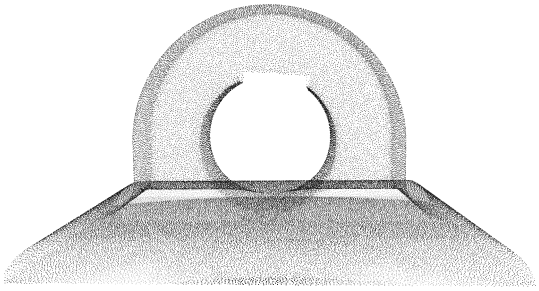


(d) Symmetric NCH

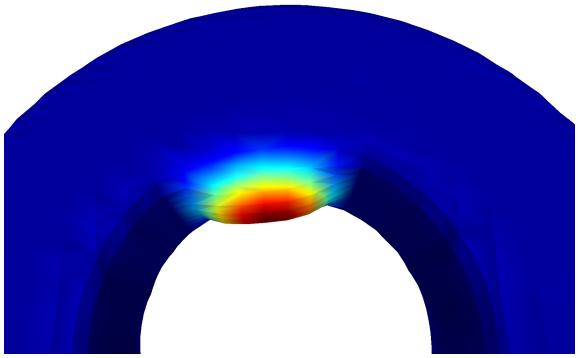


(e) Poisson

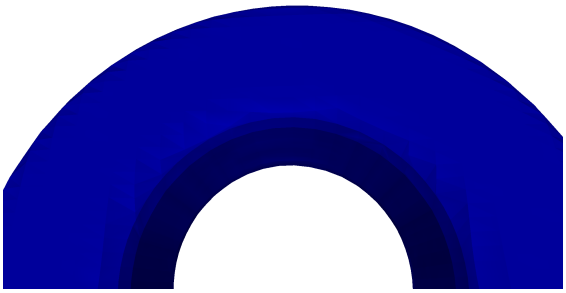
Fig. 4.37: Reconstruction of the Key Eye Pad with a piece of its corner removed. The sample size is 40k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions.



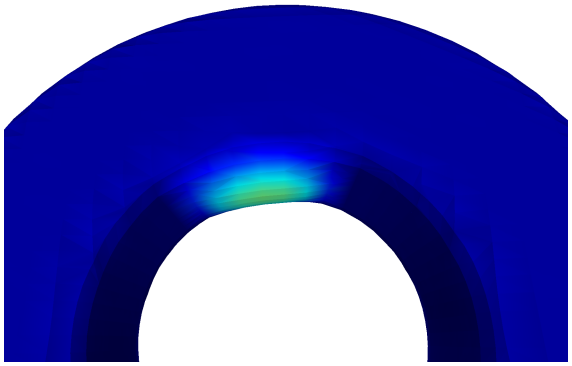
(a) Point Cloud



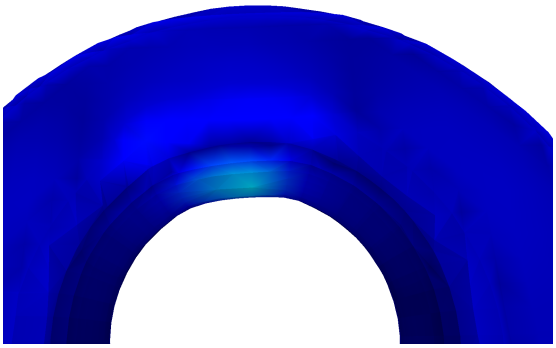
(b) Inner NCH



(c) Outer NCH

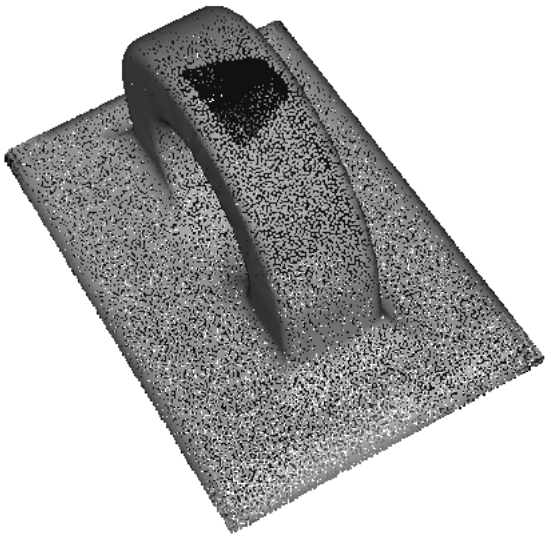


(d) Symmetric NCH

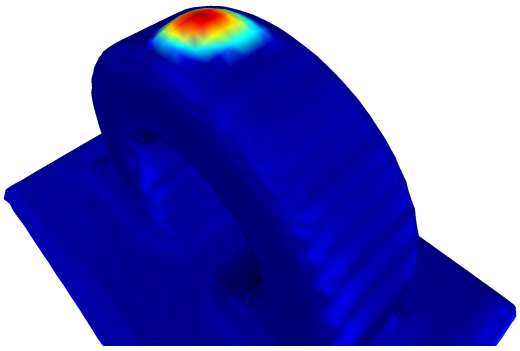


(e) Poisson

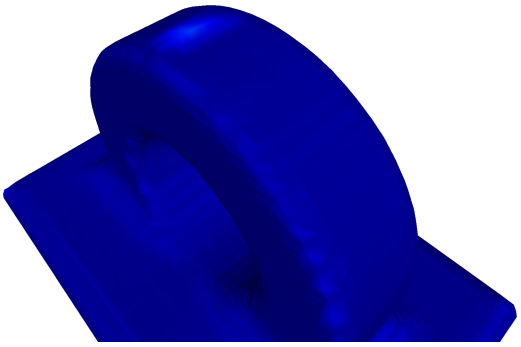
Fig. 4.38: Reconstruction of the Key Eye Pad with a part of its arc removed. The sample size is 40k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions.



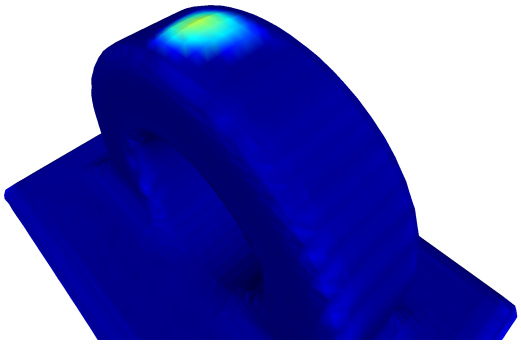
(a) Point Cloud



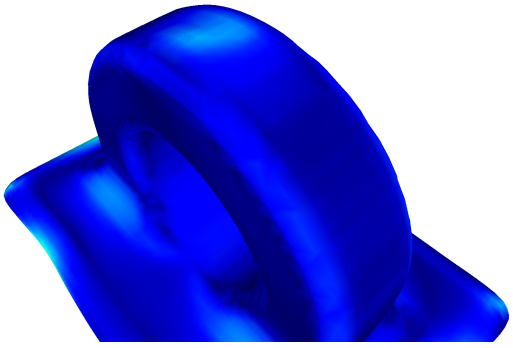
(b) Inner NCH



(c) Outer NCH



(d) Symmetric NCH



(e) Poisson

Fig. 4.39: Reconstruction of the Key Eye Pad with the upper part of its arc removed. The sample size is 40k, and the reconstruction was performed using SP and MC with a resolution of 50 in all directions.

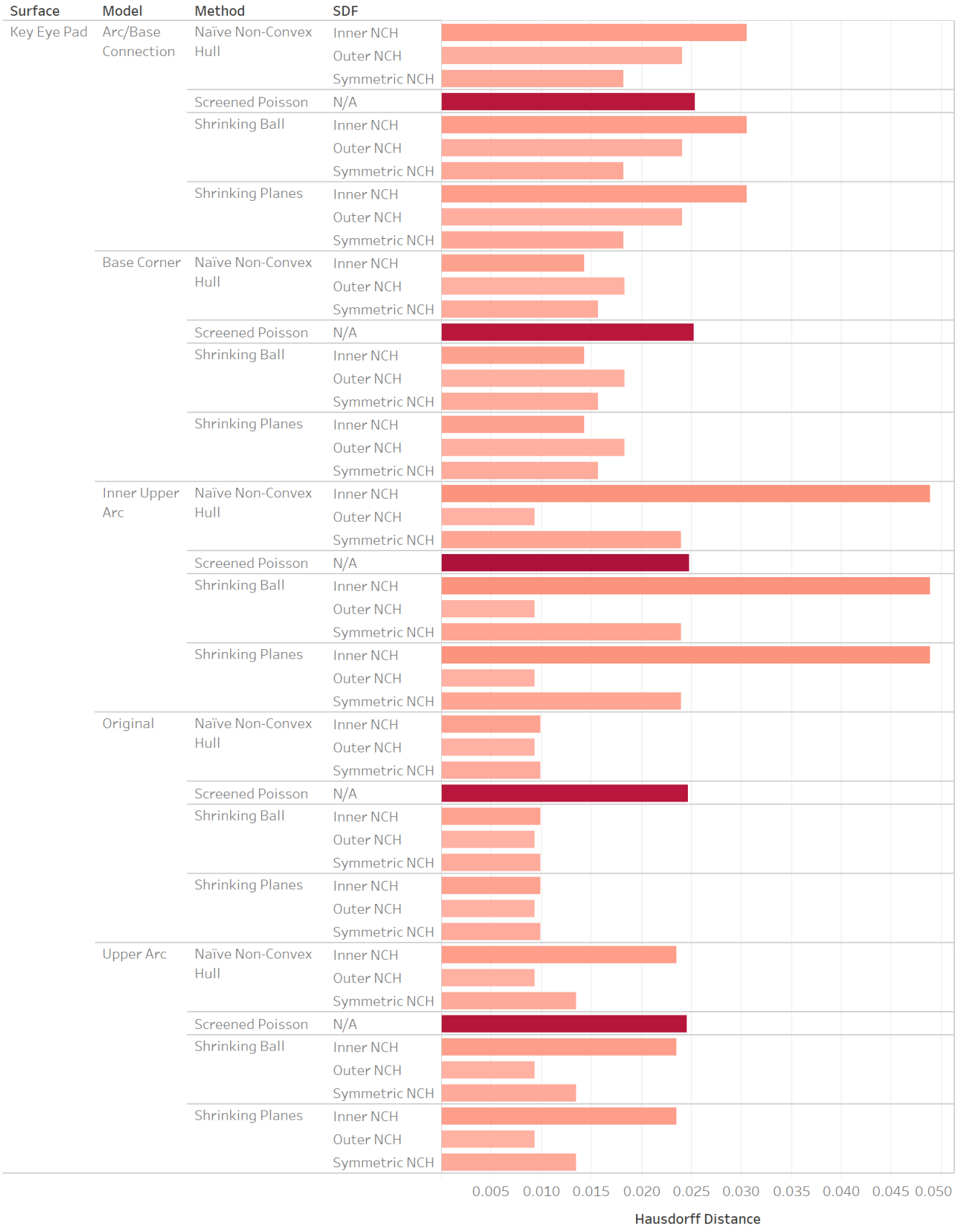


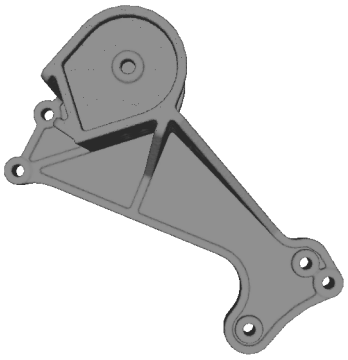
Fig. 4.40: Hausdorff Distance of the Key Eye Pad reconstruction against the reference mesh, for each change produced. Color-coded using a White-Red color palette, where White coincides with 0, and Red with the maximum value, the color is assigned as per the Average Hausdorff Distance.

4.3.4 Near Real Scans

As a final qualitative evaluation, we will compare the reconstruction of SP and SB against Screened Poisson, a state of the art method, in near real life point cloud scans produced by Berger et al. [Ber+13], using a model of a 3D scanner; the reference point clouds are shown in fig. 4.41, all others shown from here on are reconstructions from the aforementioned models. We will reconstruct many of these point clouds and compare the reconstruction quality in between methods visually. The point clouds are large in size by themselves, so we will not use NNCH for this comparison, just SP. Since these point clouds are simulations of real life scans, they also contain significant non-uniform sampling.

Due to the behavior of Poisson during the previous missing data experiments, we decided to reconstruct these figures with a depth of 10 instead of the default.

(a) The Dancing Children reference point cloud, 468k points.



(b) The Daratech reference point cloud, 245k points.

(c) The Gargoyle reference point cloud, 481k points.

(d) The Lord Quasimoto reference point cloud, 350k points.

Fig. 4.41: Reference point clouds as provided by Berger et al. [Ber+13].

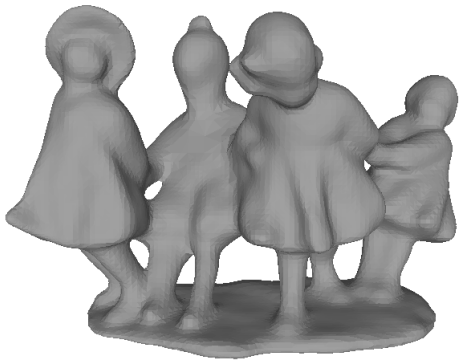
In fig. 4.42, we show a reconstruction of one of the point clouds for Dancing Children. Notice the significant amount of data missing from the body of the second from the left child, and at the legs of the children. The Poisson version is clearly overly-smoothed (details in the hair are lost, for example), but manages to produce a realistic-looking replacement for the missing data. In the Inner NCH case, the algorithm did as usual and placed multiple large spheres, while the Outer NCH did exactly the same, just outwards. Notice that in this case, there would be a significant benefit if the Symmetric NCH was used, as we would likely reconstruct something that should be slightly better than Poisson; alternatively, some smoothing on top of the Outer NCH version would provide a similar result as well.

Another Dancing Children reconstruction can be seen in fig. 4.43, in this case with significantly more points and a different missing data distribution. Notice how Inner NCH and Outer NCH reconstructions are very good in this one, even though the Inner NCH still produces the usual warts. Once again, the best SDF would likely be the Symmetric NCH, and the Outer NCH would probably resemble Poisson's reconstruction quality if it had a smoothing filter applied on top.

We also wanted to show the reconstruction of fig. 4.44. Particularly because of two things: the displayed filling of Anchor's inner side of the wheel, which the Inner NCH fills with very large balls, while the ONCH actually does the contrary (in fact, the fit is a very thin sheet). Also notice the left-side of the INCH reconstruction, which has multiple ball fits: this is because of noise and missing data on that part of the input cloud, causing overly huge spheres that go over to the incorrect side.

As last examples, we wanted to display fig. 4.45 and fig. 4.46. In these cases, both NCHs produce a good fit in the end, but could be improved to be better than Poisson if some smoothing was applied; a way to address this problem will discussed in section 5.1. In particular in the Quasimoto reconstruction of fig. 4.46, SP manages to fill the missing data much better than Poisson, which actually ends up removing Lord Quasimoto's pipe.

(a) Input point cloud. Dark areas correspond to missing data.



(b) Screened Poisson

(c) Inner NCH

(d) Outer NCH

Fig. 4.42: Reconstruction of Dancing Children with missing data. The sample size is 43k, and the reconstruction was performed using SP and MC with a resolution of 100 in all directions.

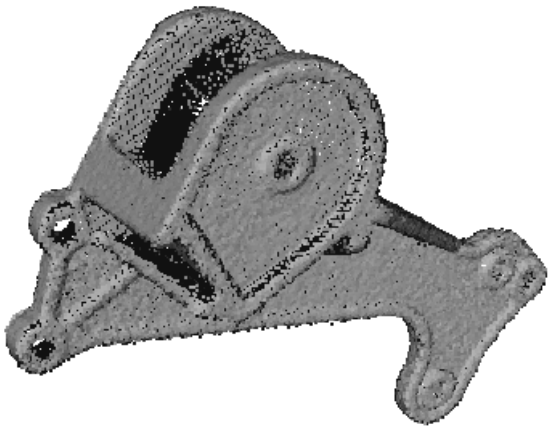
(a) Input point cloud. Dark areas correspond to missing data.

(b) Screened Poisson

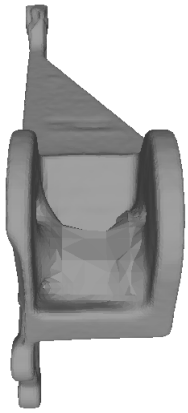
(c) Inner NCH

(d) Outer NCH

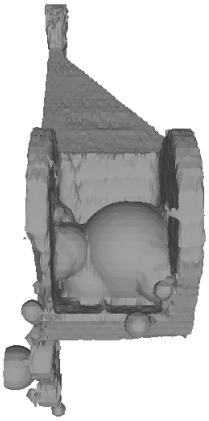
Fig. 4.43: Reconstruction of Dancing Children with missing data. The sample size is 193k, and the reconstruction was performed using SP and MC with a resolution of 100 in all directions.



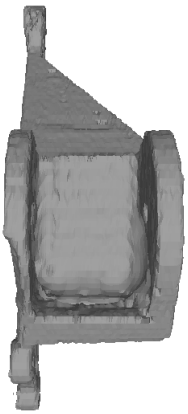
(a) Input point cloud. Dark areas correspond to missing data.



(b) Screened Poisson

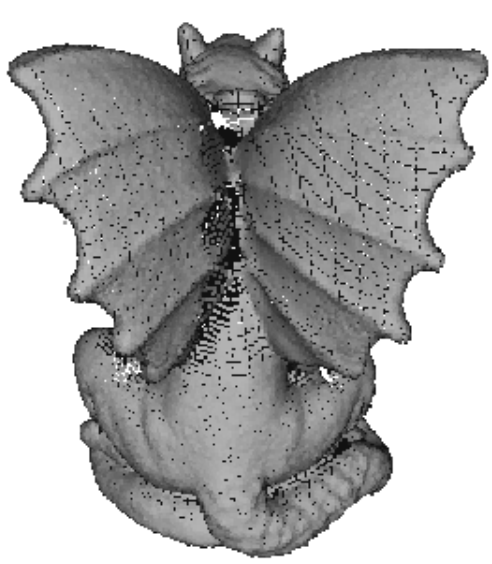


(c) Inner NCH

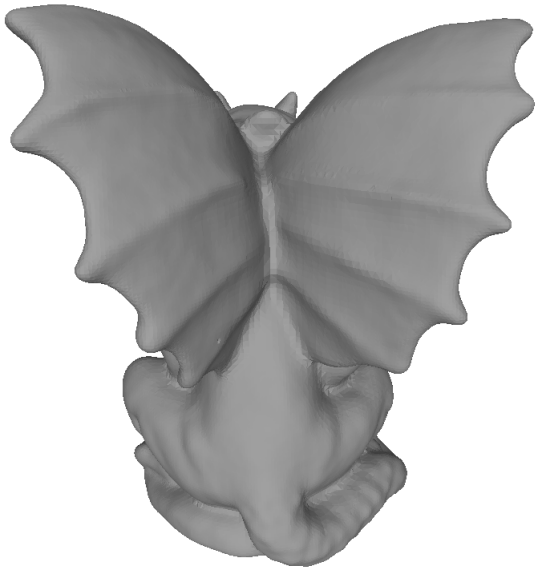


(d) Outer NCH

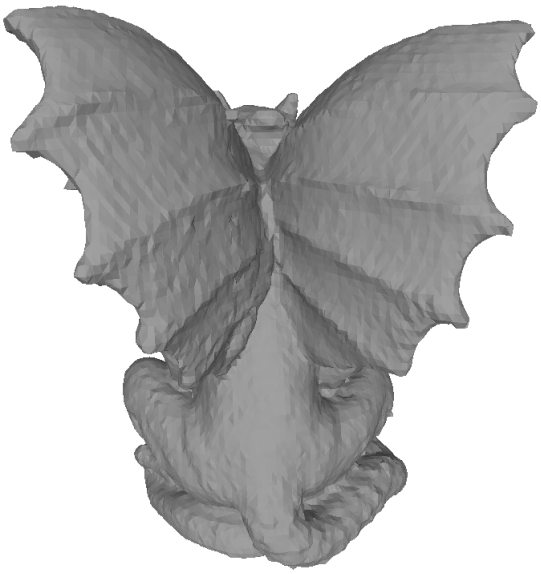
Fig. 4.44: Reconstruction of Daratech with missing data. Reconstruction was performed using SP and MC with a resolution of 100 in all directions.



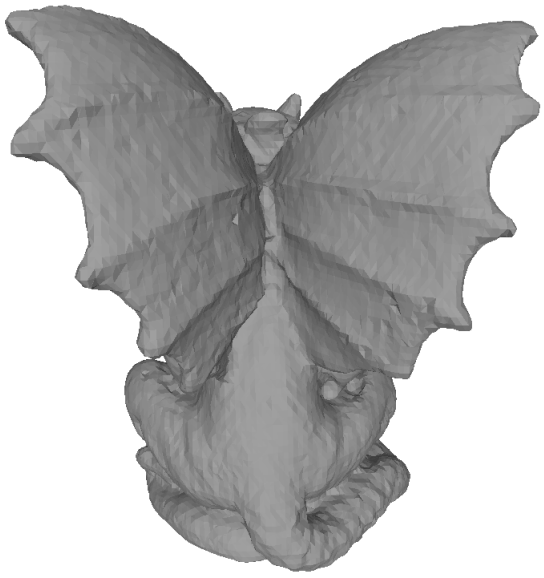
(a) Input point cloud. Dark areas correspond to missing data.



(b) Screened Poisson

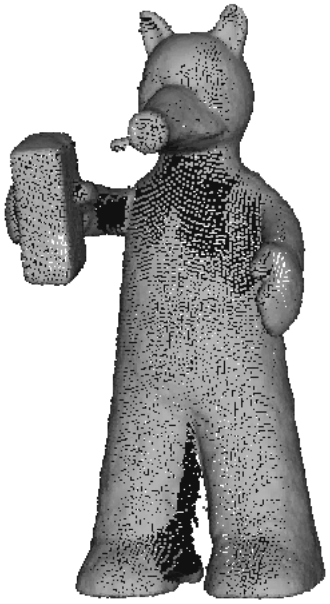


(c) Inner NCH

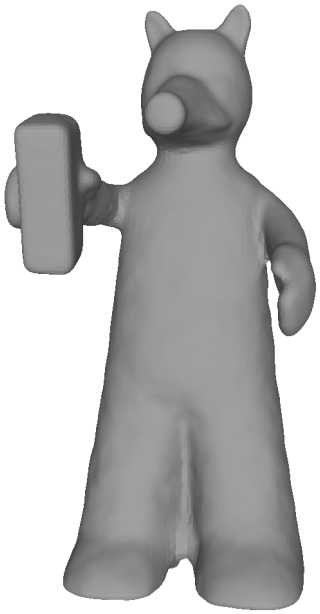


(d) Outer NCH

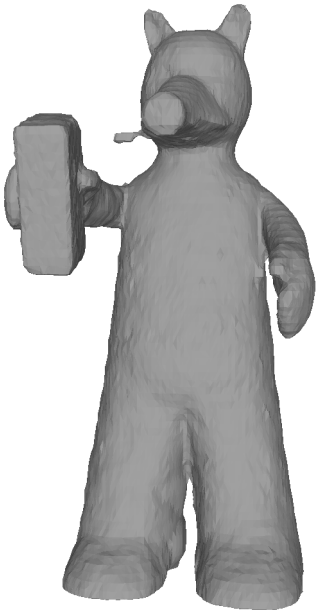
Fig. 4.45: Reconstruction of Gargoyle with missing data. Reconstruction was performed using SP and MC with a resolution of 100 in all directions.



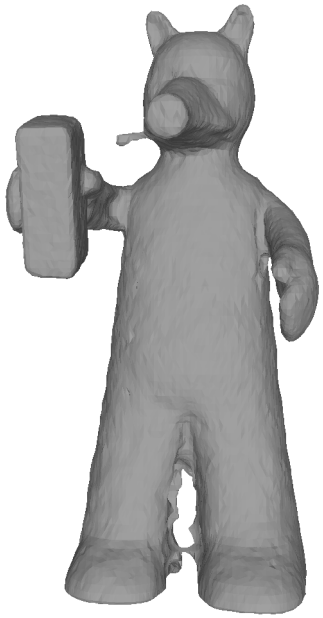
(a) Input point cloud. Dark areas correspond to missing data.



(b) Screened Poisson



(c) Inner NCH



(d) Outer NCH

Fig. 4.46: Reconstruction of Quasimoto with missing data. Reconstruction was performed using SP and MC with a resolution of 100 in all directions.

4.3.5 Performance

What has been missing from all the previous sections is any mention to what the performance of the methods presented is. We decided to concentrate such results in a single section.

We represent the NCH in memory as a sequence of floats, where each relevant subsequence is aligned to 16 bytes in order to benefit from SSE optimizations provided by the Eigen library. Concretely, we store $((x, y, z), (n_x, n_y, n_z), (\rho_i^+, \iota_i^+, \rho_i^-, \iota_i^-))$ where ι_i is a 32 bit unsigned integer equal to the index of the point that last set the radius for the current point[6]. Since each component that we need to store is either a 32 bit float or 32 bit unsigned integer, alignment is required in between, so in practice the memory layout is:

$$((x, y, z), \text{pad}_1, (n_x, n_y, n_z), \text{pad}_2, (\rho_i^+, \iota_i^+, \rho_i^-, \iota_i^-))$$

Where pad_1 and pad_2 are unused 32-bit floats. This layout allows both Eigen and the PCL to use efficient SSE optimizations in certain contexts. As a result, each point in our NCH representation takes a total size of 48 bytes; which implies that representing a 1 million point cloud's NCH requires 48 MB of memory.

Our on-disk representation of the NCH uses the well-known PLY format in its binary version, and stores the same components but compressed (i.e. without any need for the extra padding). For example, the Dragon model that we reconstructed earlier has a total of 871306 points, which makes the in-memory representation take 41.822688 MB, and the PLY representation 34 MB.

Once an NCH representation has been generated, we use it for all operations and in-place, so no further space is required for either fitting the NCH, or isoextraction.

The only additional requirements are MC's grid, and SP's KD-tree for lookups. Notice that MC's grid is of size $(K + 1)^3$ floats. For a resolution of 100 in all directions this is exactly 4 MB. This implies that the rest can be roughly attributed to the KD-Tree, and the binary representation of the program in memory. One thing to remark is that virtual memory is significantly higher than reserved memory in our case, because we have a direct dependency on Boost, the PCL, FLANN, and some other libraries; however, this is shared memory amongst all processes in the computer, and thus won't be reported.

The measurement results of running some of our point clouds using SP with MC can be seen in table 4.4. As expected, the memory requirements of SP are very low by today's

[6] When ρ_i is 0 (i.e. a plane), this is set to the maximum representable number

Model	Number of Points	Reconstruction	Isoextraction (100^3)
Dancing Children	85k	108 MB	115 MB
Anchor	120k	110 MB	140 MB
Daratech	61k	100 MB	107 MB
Lord Quasimoto	57k	101 MB	102 MB
Dragon	871k	220 MB	230 MB

Tab. 4.4: Memory usage measurements for SP with MC reconstruction on a $100 \times 100 \times 100$ grid.

standards, both for obtaining the NCH and for reconstruction.

As to the timing measurements, we decided to take the Hinge model's reference mesh and generate multiple point samples, each increasing in size, and run the full reconstruction pipeline for NNCH, SP, and SB. All measurements have been done on a Intel Core i7-6500U with 4 cores running at 2.50GHz, 16 GB of DDR3 memory, and a 1TB SSD drive; the code was compiled using the Ubuntu 18.04.2 LTS stock clang++ (version 6.0.0-1ubuntu2), Boost 1.65.1, Eigen 3.4, and PCL 1.8.1.

Programs are not parallelized on purpose to control the bottleneck for processing time; parallel implementations would of course be faster since all algorithms are embarrassingly parallel. They can also be implemented in a GPU, and in fact both Ma et al. [MBC12] and Jalba et al. [JKT13] have done so, showing fitting times of a few dozens of milliseconds.

Although we have done our best to remove all system calls from the main methods, we have been unable to do so completely: the allocation of the output point cloud is effectively part of the measurement, and inherently brings noise into the measurements, although the overhead affects all methods measure. We have used the Boost C++ Timers library to obtain time measurements.

It is important to point out that NNCH's performance does not depend on the topology of the surface being reconstructed, but SB and SP does; so different shapes could produce different performance measurements in principle. In practice, we have not seen topology being a significant performance factor for any of the shapes we reconstructed, which is why we only show one example.

The results from the measurements are shown in fig. 4.47. Notice that, as expected, for small point clouds we don't achieve a significant NCH computation gain (although there is still a 4 seconds difference on average). Also expected is the growth shown, in which our gains progressively become higher: notice how by 70k points we manage to perform the fit in roughly 24 seconds less on average. The SP algorithm is clearly slightly slower

than SB, as expected.

It is clear that the biggest bottleneck is the isoextraction part of the pipeline: a relatively low resolution reconstruction with MC takes much longer than the fitting does. This was definitely expected from the complexity computations we made earlier, and it is definitely a point for future improvement. Especially notorious is the case for the Symmetric NCH, which takes roughly an order of magnitude more than the Inner NCH and Outer NCH; this is an odd case: the code is basically the same and we would expect a 2x slowdown at most (given that the SDF computations have to be run twice). We think this has to do with some obscure compilation issue, as the code has very few modifications, we haven't been able to tell why this is happening.

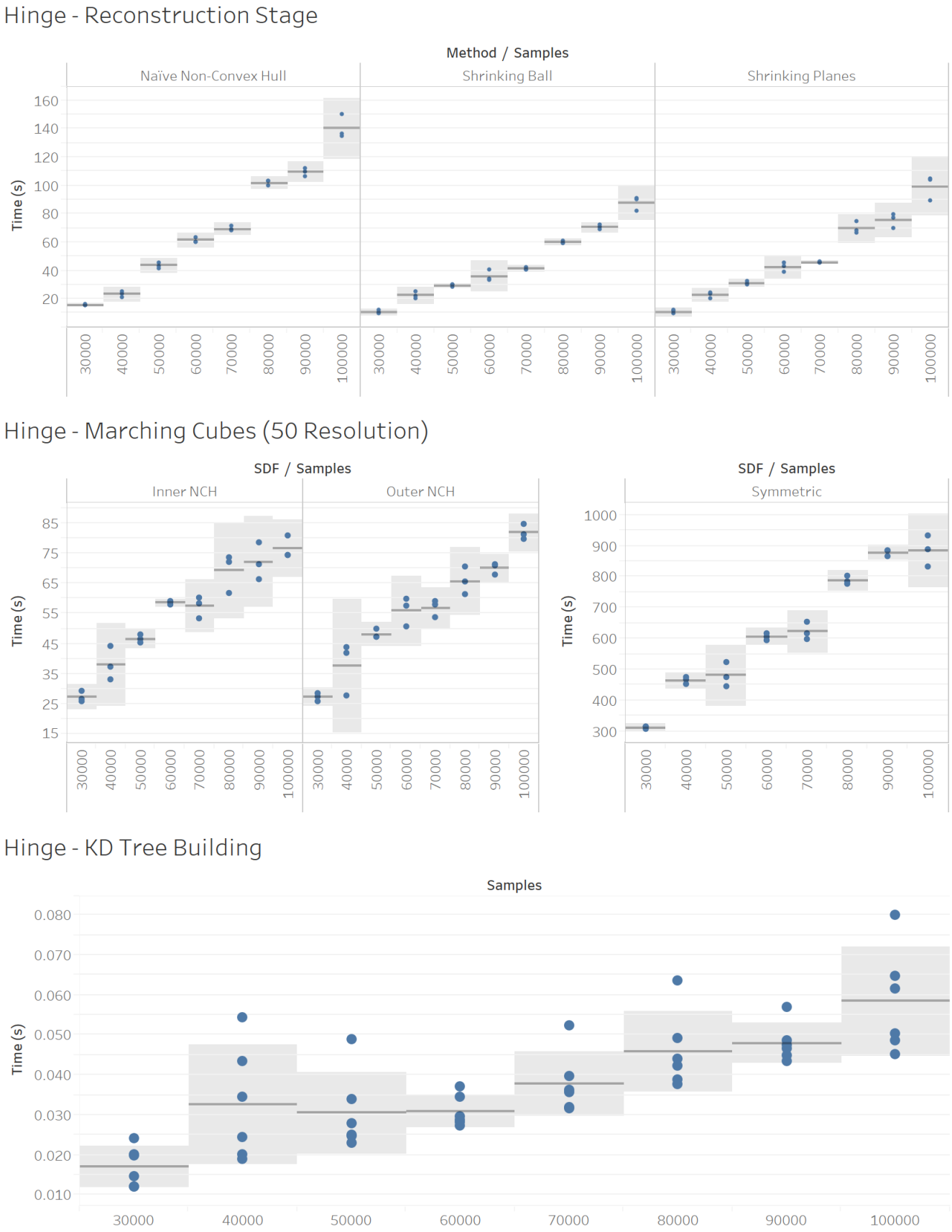


Fig. 4.47: Performance measurements, in seconds, for each part of the pipeline as the number of samples grow; three runs are shown per sample size, algorithm, and SDF. The grey lines represent the average, and the grey blocks its 95% confidence interval. The MC plots are shown separate to highlight the difference in scales.

5. CONCLUSIONS

In this work, we have presented the concept of the Medial Axis Transform (MAT), and surveyed the literature developed by several authors over the years, collecting many important properties and showing its usefulness in the formalization of surface reconstruction algorithms. More importantly, we have presented the concept of the Non-Convex Hull (NCH) as introduced by Taubin [Tau13], and given clear proofs and explanations of its properties, as well as linked it to the Medial Axis Transform (MAT) and the Power Diagram.

We have also presented the original algorithm for computing it, the Naïve Non-Convex Hull (NNCH) algorithm, as well as the Shrinking Ball (SB) algorithm for computing the MAT and its noise-resistant variant, the Denoising Shrinking Ball (DSB). Based on the aforementioned algorithms, analysis of their behavior in several models, and study of theoretical properties, we have developed the Shrinking Planes (SP) algorithm which computes an approximation to the NCH in expected linearithmic time, a significant improvement over the original quadratic algorithm. Of SP, we have proven its bounded-error reconstruction in densely sampled point clouds, and run several experiments to test both its reconstruction quality and performance in different scenarios.

Our Noise experiments have shown the superior reconstruction quality of the Outer NCH over the Inner NCH when the input is noiseless, and the preference for the Symmetric NCH when there is noise in the input. We have also demonstrated the weakness of the current NCH-based reconstruction methods against noisy point clouds, and explained why that happens. In section 5.1, we will also explain many ideas for future work to tackle this problem.

From our Missing Data experimentation, the good behavior of the Outer NCH when there is missing data over other alternatives has been a remarkable feature throughout. The Symmetric NCH's close second place in this aspect is also a direct outcome of this thesis. Indeed, after these experiments we can conclude that an approach that uses either the Inner NCH or Outer NCH on different areas would certainly be the best.

In both the Noise and Missing Data experimentation, we have shown that the SP algorithm provides an excellent approximation to the NCH as computed by the NNCH algorithm, and in particular superior to that of the SB algorithm that we adapted for the NCH.

We also performed several comparisons against Screened Poisson Surface Reconstruc-

tion, a state-of-the-art algorithm in the field, in near real world conditions by using a 3D scan simulation dataset. Out of this, the weaknesses of NCH-based methods came to be highlighted: in most examples, the individual Inner NCH or Outer NCH reconstructions produced several of its characteristic errors, which would have been prevented by using the Symmetric NCH. Nevertheless, it is quite clear that the algorithms shown in this thesis can appropriately reconstruct surfaces, albeit with some difficulty in some cases.

At last, our performance experiments showed our very small memory footprint, as well the much improved computation times of SP and SB in comparison to the Naïve Non-Convex Hull,

We consider that the original purpose of this thesis has been achieved, since all of the properties and experimentation, as well as its ease of implementation, low resource usage, and embarrassingly parallel approach, make the SP algorithm a viable candidate for future improvements that turn it into a scalable industry-strength method for surface reconstruction. Concretely, this work has definitely improved upon the original NNCH algorithm -the purpose of this thesis-, and laid the foundations for future improvements.

5.1 Future Work

This thesis is sprinkled throughout with pointers to future work that has the potential to improve the methods presented here. The purpose of this section is to gather all such material here for reference, so that the interested reader may know where to start.

5.1.1 GPU Implementation

All algorithms presented in this thesis are embarrassingly parallel, and thus very straightforward to parallelize. Ma et al. [MBC12] and Jalba et al. [JKT13] have done this previously as part of their work, and managed to get running times in the order of milliseconds for large point clouds.

The low memory usage of these algorithms also implies that a GPU implementation would allow processing of very large data sets with low overhead, such as LiDAR point clouds with billions of points. Some technical difficulties would have to be addressed, like how to partition the data on disk; fortunately, Lam [Lam16] has already worked on this topic, proposing and evaluating a few solutions.

5.1.2 Smooth Reconstruction

The Screened Poisson algorithm tends to smooth the resulting reconstruction. For many surfaces, this can be a problem, but for others it can be the key to generating a good reconstruction, for example, in fig. 3.5 a smoother reconstruction would probably fix the "bubbly" look.

In our case, we can generate a similar effect through using an alternative implementation of the CSG operations in the NCH SDF. Namely, there exist smooth replacements for the maximum and minimum functions. For example:

$$\mathcal{S}_\alpha(x_1, \ldots, x_n) = \frac{\sum_{i=1}^{n} x_i e^{\alpha x_i}}{\sum_{i=1}^{n} e^{\alpha x_i}}$$

Is known to approximate the maximum as $\alpha \to \infty$, and the minimum as $\alpha \to -\infty$. Indeed, this could be replaced in the SDF formulation for softer reconstructions. Quilez [Qui11b] does an exploratory analysis of ray-tracing SDFs using these kind of techniques with great success, as shown in fig. 5.1. For large reconstructions, this change is very likely to introduce a large overhead as it requires significant floating point operations, but it seems like a promising direction.

(a) SDF defined using the regular minimum function for union.

Source: Quilez [Qui11b]

(b) The same SDF, but using a smooth minimum implementation. Notice the union between the bridge and the snow.

Source: Quilez [Qui11b]

Fig. 5.1: A procedurally generated 3D scene. It is built using SDFs for the geometry and procedurally generated textures; the extraction is done by ray tracing rather than by isosurfacing.

5.1.3 Voronoi NCH SDF

As argued in section 3.3.4, once the radii have been determined for an NCH without planes, the resulting SDF's basis functions "domination area" correspond with a Multiplicatively

Weighted Voronoi Diagram (MW-Voronoi Diagram), which is a variant of the Voronoi Diagram that considers distances between points in exactly the same way as in lemma 3.3.5.

Therefore, if a data structure for constructing and querying the diagram fast existed, this would automatically imply that the NCH SDF could be computed as one look-up for the Inner and Outer variants, and two look-ups for the Symmetric variant. Moreover, it would allow for fast point-to-surface distance approximations using techniques such as the Taubin Distance Approximation [Tau93; Tau94].

For example, if the lookup time were decreased from $\mathcal{O}(N)$ to $\mathcal{O}(\log N)$, then MC's complexity would decrease from $\mathcal{O}((K+1)^3 N)$ to $\mathcal{O}((K+1)^3 \log N)$, which would be a very significant change.

5.1.4 Parameter Tuning

In section 3.4.3 we explained why the radius initialization heuristic by Ma et al. [MBC12] is hard to scale, as well as why the one by Jalba et al. [JKT13] is theoretically unsound. The development of better heuristics are likely to bring high returns on investment in terms of performance.

For example, we propose a similar heuristic that could be used, although no experiments have been run, which exploits the fact that the LFS is Lipschitz continuous by assuming that close points will have a similar LFS. If $\mathbf{p_j}$ is the current point, and $\mathbf{p_t}$ is the previous one, we know that:

$$|\mathbf{LFS}(\mathbf{p_j}) - \mathbf{LFS}(\mathbf{p_t})| \leq d(\mathbf{p_j}, \mathbf{p_t})$$

Which directly implies that $\mathbf{LFS}(\mathbf{p_j}) \leq \mathbf{LFS}(\mathbf{p_t}) + d(\mathbf{p_j}, \mathbf{p_t})$. Thus, we could set the new initial radius as the previous one plus the distance between the points, and this would be correct under the assumption that the current point's estimation is a good one. Notice that this heuristic does not require any sort of ordering between the points; however, it will be more effective when points are close together, as the distance will be bounded, thus producing a closer estimate to the LFS.

For example, in a noise-free ϵ-sample with an iteration ordering such that $d(\mathbf{p_j}, \mathbf{p_t}) = d(\mathbf{p_t}, \mathcal{P})$ (i.e. the next point is always the closest available), we could guarantee that:

$$|\mathbf{LFS}(\mathbf{p_j}) - \mathbf{LFS}(\mathbf{p_t})| \leq \mathcal{O}(\epsilon)\mathbf{LFS}(\mathbf{p_t})$$

Which would imply that the most that we can overshoot the true $\mathbf{LFS}(\mathbf{p_j})$ with this heuristic is indeed bounded, and furthermore controllable: taking $\epsilon \leq \frac{\tau}{\sup_{\mathbf{x}\in\partial\mathcal{S}} \mathbf{LFS}(\mathbf{x})}$ lets

us bound the overshoot by τ. Under the assumption that ϵ is sufficiently small, both expressions become equivalent -since the LFS is continuous-, and hence we are assigning to it a number that is optimal. Notice that two points could be each others closest sample, implying that the guarantee cannot be proven in general, but it does give this heuristic an interesting behavior to start with.

Aside from this, the number of points inside the MA used are taken as a fixed number in section 4.1, but there is no particular why those couldn't be variable: as the ball gets progressively smaller, it is unlikely that new points will be found to be inside, so it makes sense to decrease the amount searched for.

Another point to be optimized is the initial approximation stage: although we currently keep shrinking into convergence, it would be possible to stop earlier and let the second iterative refinement finish the job. Nevertheless, this is a difficult optimization: remember that the KD-Tree has $\mathcal{O}(N)$ complexity for the radius search.

5.1.5 Adaptive Reconstruction

In this thesis, the methods used are global: all points are used whenever estimating the parameters for any given point. They are also of fixed resolution: we always fit as many balls as we possibly can for a given shape, and have no level of detail in the reconstruction.

Both characteristics are not always in our best interest: the global nature of the methods implies that noise may be devastating for the reconstruction process, while the fixed resolution means we may incur an unnecessarily high cost when reconstructing certain surfaces, even if there is no need to do so.

Spherical representations have been used in the literature for a long time, particularly for fast object collision testing. Hubbard [Hub96] pioneered the methodology, and Bradshaw and O'Sullivan [BO04] wrote what seems to be the first Adaptive MAT approximation. It is likely for a similar approach to work reasonably well in this domain.

A much more related approach is that of Bianchi [Bia19], who implemented an algorithm using the subdivision techniques from MPU, while using NNCH at the leaves for fitting a local approximation. His approach successfully competes with Poisson Reconstruction, as shown in his thesis, and thus future work could directly replace his NNCH implementation for our SP algorithm for an increased computation speed.

As we have shown throughout our experiments, the Outer NCH and Inner NCH generally complement each other: where one produces a very good fit, the other does not, and vice versa. An adaptive approach that used one or the other depending on the local surface characteristics could be a very good intermediate method, and probably would be

better than all SDFs used in this thesis. Notice that nothing would change in the fitting method, only in the isosurfacing stage; so only reconstruction tests would be needed.

5.1.6 Adaptive IsoExtraction

Taubin [Tau13] originally implemented the NNCH algorithm for fitting, along with DMC for isosurface extraction.

The most time-consuming aspect of our implementation is the surface isoextraction, as the rest of the pipeline is very fast. Thus, we could greatly benefit from an improved isosurfacing implementation.

The NCH has a lot of structure, which could be exploited for faster isoextraction. For example, we know that the only atoms possible are balls or planes; this implies that the MC grid can be computed faster: we can bound the contribution of a ball by its radius. Thus, the grid computation would just iterate through all balls, adding the contribution to each vertex as appropriate, and then loop through all planes, adding the contribution to all vertices in that particular case.

5.1.7 Simplification

After the NCH has been fit, we have a very rich representation of the surface. As we have shown in our experimentation, the NCH representation can often be very redundant, even for simple shapes such as spheres and cubes.

NCH simplification techniques would allow us to remove the redundancy in the representation, and hence decrease computation times of subsequent operations. Being a superset of the MAT, the NCH is a very rich surface representation, and puts a lot of information in our hands with which to build a good simplification algorithm: we can have local curvature bounds, LFS estimation, among others. Furthermore, simplification techniques for the MAT can be translated for the NCH when looking only at non-plane fits. There is a huge body of literature on this topic, so we will only cite here works that caught our eye and we think lead in interesting directions.

Foskey et al. [FLM03] have worked on approximating the θ-SMA and the λ-MAT [CL05]; both of which are subsets of the MAT with better stability properties. Indeed, the stability guarantees also imply that these subsets are also good noise removal mechanisms. As we mentioned earlier, Peters [Pet18; Pet14d; Pet14a] developed the DSB algorithm based on Foskey's work with the θ-SMA. There are three possible future work routes in this direction:

- We can adapt the DSB algorithm to fit the NCH with SP's additional heuristic.
- As mentioned during our experiments, we found the DSB algorithm to be extremely sensitive to its parameters, and to require a good deal of tweaking to achieve the desired operation. Developing techniques for choosing the parameters effectively would be very useful.
- No adaptation of the SB for approximating the λ-MAT has been produced as far as we know, so any work in this direction would certainly be interesting.

In the same vein, Peters [Pet18; Pet14b; Pet14c] worked in LFS-sensitive point decimation, which can also be performed directly from the MAT representation. We have not tested this algorithm in particular for the surface reconstruction problem, but it may prove to be a good way to retain only the most important features in a surface.

It can be proven that $2\rho_i$ is an upper bound for the local curvature at a point [SPW96; She95]. The LFS and curvature are data that have been used previously for simplification of both meshes and point clouds, and could likewise be used to guide an NCH simplification algorithm.

5.1.8 Noise-resistant NCH

Our experimentation revealed the severe problems that the NCH fitting algorithms we have implemented face in the presence of noise: even the slightest amount would cripple a reconstruction. This is a major road block in the path to a state-of-the-art surface reconstruction algorithm. Aside from the denoising heuristic for DSB, and the ideas for simplification that we presented in the previous section, there are alternative approaches we think could work for this problem. For these, we will instead assume that we have an NCH fit already, and attempt to improve its robustness in the presence of noise.

As Choi et al. [CCM97] showed, very small deformations similar to the ones that could be generated by noise generate branches (in 2D, sheets in 3D) in the MAT. We propose a method based on the Lipschitz continuity of the LFS for this. The MAT under noise would have additional branches, so if we assume that noise is unstructured (i.e. does not have a pattern to it), we would have highly different LFS between near by spheres, which would break the continuity condition.

This automatically presents us with the opportunity to basically attempt to minimize the distance of a set of points to the border of a set of spheres (i.e. ensure that points are at the border or outside), while using a regularization term that helps keep the radius

difference as low as possible. Several methods for Least Squares Sphere and Ellipse fitting are covered in Gander et al. [GGS94] and more recently in Umbach and Jones [UJ03].

These kind of ideas have been applied in one way or another for the same or tangential purposes already. For example, Li et al. [Li+15] perform MAT approximation using a Least-Squares fit over a mesh representation, and Taubin [Tau12] produce a state-of-the-art smooth surface reconstruction method through minimization of a cost function using the Conjugate Gradient method.

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

[Ale+01] Marc Alexa et al. "Point Set Surfaces". In: *Proceedings of the Conference on Visualization '01*. VIS '01. San Diego, California: IEEE Computer Society, 2001, pp. 21–28. ISBN: 0-7803-7200-X. URL: http://dl.acm.org/citation.cfm?id=601671.601673.

[AB98] Nina Amenta and Marshall Bern. "Surface Reconstruction by Voronoi Filtering". In: *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*. SCG '98. Minneapolis, Minnesota, USA: ACM, 1998, pp. 39–48. ISBN: 0-89791-973-4. DOI: 10.1145/276884.276889. URL: http://doi.acm.org/10.1145/276884.276889.

[ACK01] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. "The Power Crust, Unions of Balls, and the Medial Axis Transform". In: *Comput. Geom. Theory Appl.* 19.2-3 (July 2001), pp. 127–153. ISSN: 0925-7721. DOI: 10.1016/S0925-7721(01)00017-7. URL: http://dx.doi.org/10.1016/S0925-7721(01)00017-7.

[AK00] Nina Amenta and Ravi Krishna Kolluri. "Accurate and Efficient Unions of Balls". In: *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*. SCG '00. Clear Water Bay, Kowloon, Hong Kong: ACM, 2000, pp. 119–128. ISBN: 1-58113-224-7. DOI: 10.1145/336154.336193. URL: http://doi.acm.org/10.1145/336154.336193.

[ABE09] Dominique Attali, Jean-Daniel Boissonnat, and Herbert Edelsbrunner. "Stability and Computation of Medial Axes - a State-of-the-Art Report". In: *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*. Ed. by Torsten Möller, Bernd Hamann, and Robert D. Russell. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 109–125. ISBN: 978-3-540-49926-8. DOI: 10.1007/b106657_6. URL: https://doi.org/10.1007/b106657_6.

[Ber+08] Mark de Berg et al. *Computational Geometry: Algorithms and Applications*. 3rd ed. Santa Clara, CA, USA: Springer-Verlag TELOS, 2008. ISBN: 3540779736, 9783540779735.

[BS12] Matthew Berger and Claudio T. Silva. "Medial Kernels". In: *Comput. Graph. Forum* 31.2pt4 (May 2012), pp. 795–804. ISSN: 0167-7055. DOI: 10.1111/j.1467-8659.2012.03060.x. URL: http://dx.doi.org/10.1111/j.1467-8659.2012.03060.x.

[Ber+13] Matthew Berger et al. "A Benchmark for Surface Reconstruction". In: *ACM Trans. Graph.* 32.2 (Apr. 2013), 20:1–20:17. ISSN: 0730-0301. DOI: 10.1145/2451236.2451246. URL: http://doi.acm.org/10.1145/2451236.2451246.

[Ber+17] Matthew Berger et al. "A Survey of Surface Reconstruction from Point Clouds". In: *Comput. Graph. Forum* 36.1 (Jan. 2017), pp. 301–329. ISSN: 0167-7055. DOI: 10.1111/cgf.12802. URL: https://doi.org/10.1111/cgf.12802.

[Bia19] Shai Bianchi. "Estimación de la cápsula no-convexa de nubes de puntos por partición de la unidad". MA thesis. Universidad de Buenos Aires, Feb. 2019.

[Blu67] Harry Blum. "A Transformation for Extracting New Descriptors of Shape". In: *Models for the Perception of Speech and Visual Form* (1967). Ed. by Weiant Wathen-Dunn, pp. 362–380. URL: http://pageperso.lif.univ-mrs.fr/%5C~%7B%7Dedouard.thiel/rech/1967-blum.pdf.

[Bor+09] Alex Bordignon et al. "Scale-space for union of 3d balls". In: *Sibgrapi 2009 (XXII Brazilian Symposium on Computer Graphics and Image Processing)*. Rio de Janeiro, RJ: IEEE, Oct. 2009, pp. 9–16. DOI: 10.1109/Sibgrapi.2009.9. URL: %5Curl%7Bhttp://thomas.lewiner.org/pdfs/medialmove3d_sibgrapi.pdf%7D.

[BO04] Gareth Bradshaw and Carol O'Sullivan. "Adaptive Medial-axis Approximation for Sphere-tree Construction". In: *ACM Trans. Graph.* 23.1 (Jan. 2004), pp. 1–26. ISSN: 0730-0301. DOI: 10.1145/966131.966132. URL: http://doi.acm.org/10.1145/966131.966132.

[CT12] Fatih Calakli and Gabriel Taubin. "SSD-C: Smooth Signed Distance Colored Surface Reconstruction". In: *Expanding the Frontiers of Visual Analytics and Visualization*. Ed. by John Dill et al. London: Springer London, 2012, pp. 323–338. ISBN: 978-1-4471-2804-5. DOI: 10.1007/978-1-4471-2804-5_18. URL: https://doi.org/10.1007/978-1-4471-2804-5_18.

[Căr+06] Bogdan Cărbunar et al. "Redundancy and Coverage Detection in Sensor Networks". In: *ACM Trans. Sen. Netw.* 2.1 (Feb. 2006), pp. 94–128. ISSN: 1550-4859. DOI: 10.1145/1138127.1138131. URL: http://doi.acm.org/10.1145/1138127.1138131.

[CL05] Frédéric Chazal and André Lieutier. "The Lambda-Medial Axis". In: *Graph. Models* 67.4 (July 2005), pp. 304–331. ISSN: 1524-0703. DOI: 10.1016/j.gmod.2005.01.002. URL: http://dx.doi.org/10.1016/j.gmod.2005.01.002.

[CCM97] Hyeong In Choi, Sung Woo Choi, and Hwan Pyo Moon. "Mathematical Theory Of Medial Axis Transform". In: *Pacific J. Math* 181 (1997), pp. 57–88.

[CRS98] P. Cignoni, C. Rocchini, and R. Scopigno. "Metro: Measuring Error on Simplified Surfaces". In: *Computer Graphics Forum* 17.2 (1998), pp. 167–174. DOI: 10.1111/1467-8659.00236. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00236. URL: https://doi.org/10.1111/1467-8659.00236.

[Cig+08] Paolo Cignoni et al. "MeshLab: an Open-Source Mesh Processing Tool". In: *Eurographics Italian Chapter Conference*. Ed. by Vittorio Scarano, Rosario De Chiara, and Ugo Erra. The Eurographics Association, 2008. ISBN: 978-3-905673-68-5. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[CCS12] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. "Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes". In: *IEEE Transaction on Visualization and Computer Graphics* 18.6 (2012). http://doi.ieeecomputersociety pp. 914–924. URL: http://vcg.isti.cnr.it/Publications/2012/CCS12.

[CKM99] Tim Culver, John Keyser, and Dinesh Manocha. "Accurate Computation of the Medial Axis of a Polyhedron". In: *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*. SMA '99. Ann Arbor, Michigan, USA: ACM, 1999, pp. 179–190. ISBN: 1-58113-080-5. DOI: 10.1145/304012.304030. URL: http://doi.acm.org/10.1145/304012.304030.

[DRF12] T. Delamé, C. Roudet, and D. Faudot. "From a Medial Surface To a Mesh". In: *Computer Graphics Forum* 31.5 (2012), pp. 1637–1646. DOI: 10.1111/j.1467-8659.2012.03169.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2012.03169.x. URL: https://doi.org/10.1111/j.1467-8659.2012.03169.x.

[Dey06] Tamal K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis (Cambridge Monographs on Applied and Computational Mathematics)*. New York, NY, USA: Cambridge University Press, 2006. ISBN: 0521863708.

[DZ02] Tamal K. Dey and Wulue Zhao. “Approximate Medial Axis As a Voronoi Subcomplex”. In: *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications*. SMA ’02. Saarbrücken, Germany: ACM, 2002, pp. 356–366. ISBN: 1-58113-506-8. DOI: 10.1145/566282.566333. URL: http://doi.acm.org/10.1145/566282.566333.

[Dob] Adam Dobrin. *A Review of Properties and Variations of Voronoi Diagrams*. URL: https://www.whitman.edu/Documents/Academics/Mathematics/dobrinat.pdf.

[FLM03] Mark Foskey, Ming C. Lin, and Dinesh Manocha. “Efficient Computation of a Simplified Medial Axis”. In: *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*. SM ’03. Seattle, Washington, USA: ACM, 2003, pp. 96–107. ISBN: 1-58113-706-0. DOI: 10.1145/781606.781623. URL: http://doi.acm.org/10.1145/781606.781623.

[GGS94] Walter Gander, Gene H. Golub, and Rolf Strebel. “Least-Squares Fitting of Circles and Ellipses”. In: *BIT Numerical Mathematics* 34.4 (Dec. 1994), pp. 558–578. ISSN: 1572-9125. DOI: 10.1007/BF01934268. URL: https://doi.org/10.1007/BF01934268.

[GK04] Peter Giblin and Benjamin B. Kimia. “A Formal Classification of 3D Medial Axis Points and Their Local Geometry”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 26.2 (Jan. 2004), pp. 238–251. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2004.1262192. URL: https://doi.org/10.1109/TPAMI.2004.1262192.

[GG07] Gaël Guennebaud and Markus Gross. “Algebraic Point Set Surfaces”. In: *ACM Trans. Graph.* 26.3 (July 2007). ISSN: 0730-0301. DOI: 10.1145/1276377.1276406. URL: http://doi.acm.org/10.1145/1276377.1276406.

[Hei14] Juha Heinonen. *Lectures on Lipschitz Analysis*. Aug. 2014. URL: http://www.math.jyu.fi/research/reports/rep100.pdf.

[Hof90] Christoph M. Hoffmann. “How to Construct the Skeleton of CSG Objects”. In: 1990.

[Hop+92] Hugues Hoppe et al. “Surface Reconstruction from Unorganized Points”. In: *SIGGRAPH Comput. Graph.* 26.2 (July 1992), pp. 71–78. ISSN: 0097-8930. DOI: 10.1145/142920.134011. URL: http://doi.acm.org/10.1145/142920.134011.

[Hub96] Philip M. Hubbard. "Approximating Polyhedra with Spheres for Time-critical Collision Detection". In: *ACM Trans. Graph.* 15.3 (July 1996), pp. 179–210. ISSN: 0730-0301. DOI: 10.1145/231731.231732. URL: http://doi.acm.org/10.1145/231731.231732.

[JKT13] A. C. Jalba, J. Kustra, and A. C. Telea. "Surface and Curve Skeletonization of Large 3d Models on the Gpu". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.6 (June 2013), pp. 1495–1508. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.212. URL: https://doi.org/10.1109/TPAMI.2012.212.

[Ju+02] Tao Ju et al. "Dual Contouring of Hermite Data". In: *ACM Trans. Graph.* 21.3 (July 2002), pp. 339–346. ISSN: 0730-0301. DOI: 10.1145/566654.566586. URL: http://doi.acm.org/10.1145/566654.566586.

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson Surface Reconstruction". In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. SGP '06. Cagliari, Sardinia, Italy: Eurographics Association, 2006, pp. 61–70. ISBN: 3-905673-36-3. URL: http://dl.acm.org/citation.cfm?id=1281957.1281965.

[KH13] Michael Kazhdan and Hugues Hoppe. "Screened Poisson Surface Reconstruction". In: *ACM Trans. Graph.* 32.3 (July 2013), 29:1–29:13. ISSN: 0730-0301. DOI: 10.1145/2487228.2487237. URL: http://doi.acm.org/10.1145/2487228.2487237.

[Lam16] Marco Lam. "Creating the medial axis transform for billions of LiDAR points using a memory efficient method". MA thesis. MSc thesis in Geomatics, Delft University of Technology, 2016. URL: http://resolver.tudelft.nl/uuid:2452566b-058a-4fde-aa0e-e300421bb8b2/.

[LT09] Douglas Lanman and Gabriel Taubin. "Build Your Own 3D Scanner: 3D Photograhy for Beginners". In: *SIGGRAPH '09: ACM SIGGRAPH 2009 courses*. New Orleans, LA USA: ACM, 2009, pp. 1–87.

[Lew+05] Thomas Lewiner et al. "Curvature motion for union of balls". In: *Sibgrapi 2005 (XVIII Brazilian Symposium on Computer Graphics and Image Processing)*. Natal, RN: IEEE, Oct. 2005, pp. 47–54. DOI: 10.1109/SIBGRAPI.2005.24. URL: %5Curl%7Bhttp://thomas.lewiner.org/pdfs/medialmove_sibgrapi.pdf%7D.

[Li+15] Pan Li et al. "Q-MAT: Computing Medial Axis Transform By Quadratic Error Minimization". In: *ACM Trans. Graph.* 35.1 (Dec. 2015), 8:1–8:16. ISSN: 0730-0301. DOI: 10.1145/2753755. URL: http://doi.acm.org/10.1145/2753755.

[LC87] William E. Lorensen and Harvey E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: *SIGGRAPH Comput. Graph.* 21.4 (Aug. 1987), pp. 163–169. ISSN: 0097-8930. DOI: 10.1145/37402.37422. URL: http://doi.acm.org/10.1145/37402.37422.

[MBC12] Jaehwan Ma, Sang Won Bae, and Sunghee Choi. "3d Medial Axis Point Approximation Using Nearest Neighbors and the Normal Field". In: *The Visual Computer* 28.1 (Jan. 2012), pp. 7–19. ISSN: 1432-2315. DOI: 10.1007/s00371-011-0594-7. URL: https://doi.org/10.1007/s00371-011-0594-7.

[Nie04] G. M. Nielson. "Dual Marching Cubes". In: *IEEE Visualization 2004*. Oct. 2004, pp. 489–496. DOI: 10.1109/VISUAL.2004.28.

[Oht+03] Yutaka Ohtake et al. "Multi-level Partition of Unity Implicits". In: *ACM Trans. Graph.* 22.3 (July 2003), pp. 463–470. ISSN: 0730-0301. DOI: 10.1145/882262.882293. URL: http://doi.acm.org/10.1145/882262.882293.

[Pet14a] Ravi Peters. *Approximating the Medial Axis Transform of LiDAR point clouds.* Poster at the Lorentz workshop on geometric algorithms in the field. June 2014. URL: https://3d.bk.tudelft.nl/rypeters/pdfs/Lorentz_poster_web.pdf.

[Pet14b] Ravi Peters. *Feature aware digital surface model analysis and generalization based on the 3D medial axis transform.* PhD research proposal GISt Report No. 65. Delft University of Technology, 2014. URL: https://3d.bk.tudelft.nl/rypeters/pdfs/14phdproposal.pdf.

[Pet14c] Ravi Peters. *Feature-aware LiDAR point cloud simplification.* Poster at the GeoBuzz conference. Nov. 2014. URL: https://3d.bk.tudelft.nl/rypeters/pdfs/GeoBuzz_poster_web.pdf.

[Pet14d] Ravi Peters. *Towards Medial Axis-based simplification of LiDAR point clouds.* Presentation at the iQumulus workshop (Cardiff, United Kingdom). July 2014. URL: https://3d.bk.tudelft.nl/rypeters/pdfs/14_iqmulus_workshop.pdf.

[Pet18] Ravi Peters. "Geographical point cloud modelling with the 3D medial axis transform". ISBN: 978-94-6186-899-2. PhD thesis. Delft University of Technology, Mar. 2018. URL: https://doi.org/10.4233/uuid:f3a5f5af-ea54-40ba-8702-e193a087f243.

[Qui11a] Inigo Quilez. *Distance Functions*. 2011. URL: https://www.iquilezles.org/www/articles/distfunctions/distfunctions.htm (visited on 01/31/2019).

[Qui11b] Inigo Quilez. *Smooth Minimum*. 2011. URL: https://www.iquilezles.org/www/articles/smin/smin.htm (visited on 03/07/2019).

[RC11] R. B. Rusu and S. Cousins. "3D is here: Point Cloud Library (PCL)". In: *2011 IEEE International Conference on Robotics and Automation*. May 2011, pp. 1–4. DOI: 10.1109/ICRA.2011.5980567.

[SPW96] Evan C Sherbrooke, Nicholas M Patrikalakis, and Franz-Erich Wolter. "Differential and Topological Properties of Medial Axis Transforms". In: *Graphical Models and Image Processing* 58.6 (1996), pp. 574–592. ISSN: 1077-3169. DOI: https://doi.org/10.1006/gmip.1996.0047. URL: http://www.sciencedirect.com/science/article/pii/S1077316996900477.

[She95] Evan Conway Sherbrooke. "3D Shape Interrogation by Medial Axis Transform". PhD thesis. Cambridge, MA, USA, 1995. URL: http://hdl.handle.net/1721.1/11450.

[SP08] Kaleem Siddiqi and Stephen Pizer. *Medial Representations: Mathematics, Algorithms and Applications*. 1st. Springer Publishing Company, Incorporated, 2008. ISBN: 1402086571.

[SKS12] Svetlana Stolpner, Paul Kry, and Kaleem Siddiqi. "Medial Spheres for Shape Approximation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.6 (June 2012), pp. 1234–1240. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2011.254. URL: http://dx.doi.org/10.1109/TPAMI.2011.254.

[Tag+16] Andrea Tagliasacchi et al. "3D Skeletons: A State-of-the-Art Report". In: *Computer Graphics Forum* 35.2 (2016), pp. 573–597. DOI: 10.1111/cgf.12865. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12865. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12865.

[TH03] R. Tam and W. Heidrich. “Shape simplification based on the medial axis transform”. In: *IEEE Visualization, 2003. VIS 2003.* Oct. 2003, pp. 481–488. DOI: 10.1109/VISUAL.2003.1250410.

[Tau93] Gabriel Taubin. “An Accurate Algorithm for Rasterizing Algebraic Curves”. In: *Proceedings on the Second ACM Symposium on Solid Modeling and Applications.* SMA ’93. Montreal, Quebec, Canada: ACM, 1993, pp. 221–230. ISBN: 0-89791-584-4. DOI: 10.1145/164360.164427. URL: http://doi.acm.org/10.1145/164360.164427.

[Tau94] Gabriel Taubin. “Distance Approximations for Rasterizing Implicit Curves”. In: *ACM Trans. Graph.* 13.1 (Jan. 1994), pp. 3–42. ISSN: 0730-0301. DOI: 10.1145/174462.174531. URL: http://doi.acm.org/10.1145/174462.174531.

[Tau12] Gabriel Taubin. “Smooth Signed Distance Surface Reconstruction and Applications”. In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications.* Ed. by Luis Alvarez et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 38–45. ISBN: 978-3-642-33275-3.

[Tau13] Gabriel Taubin. “Non-Convex Hull Surfaces”. In: *SIGGRAPH Asia 2013 Technical Briefs.* SA ’13. Hong Kong, Hong Kong: ACM, 2013, 2:1–2:4. ISBN: 978-1-4503-2629-2. DOI: 10.1145/2542355.2542358. URL: http://doi.acm.org/10.1145/2542355.2542358.

[UJ03] D. Umbach and K. N. Jones. “A Few Methods for Fitting Circles To Data”. In: *IEEE Transactions on Instrumentation and Measurement* 52.6 (Dec. 2003), pp. 1881–1885. ISSN: 0018-9456. DOI: 10.1109/TIM.2003.820472. URL: https://doi.org/10.1109/TIM.2003.820472.

[Wen13] Rephael Wenger. “Isosurfaces: Geometry, Topology, and Algorithms”. In: (Jan. 2013). Ed. by A. K. Peters, p. 488. DOI: 10.1201/b15025.

[Zhu+14] Yanshu Zhu et al. “Computing a Compact Spline Representation of the Medial Axis Transform of a 2D Shape”. In: *Graph. Models* 76.5 (Sept. 2014), pp. 252–262. ISSN: 1524-0703. DOI: 10.1016/j.gmod.2014.03.007. URL: http://dx.doi.org/10.1016/j.gmod.2014.03.007.