Surface Reconstruction for Computed Tomography Volumes

Auteur : Douhard, Robin
Promoteur(s) : Geuzaine, Christophe
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil en informatique, à finalité spécialisée en "computer systems security"
Année académique : 2024-2025
URI/URL : http://hdl.handle.net/2268.2/23375

UNIVERSITE DE LIEGE
FACULTE DES SCIENCES APPLIQUEES

Surface Reconstruction for Computed Tomography Volumes

Thesis presented to obtain the degree of :
Master of Science in Computer Science and Engineering

Author
Robin Douhard

Supervisors
Prof. Christophe Geuzaine
Dr. Vincent Libertiaux

Academic year: **2024 - 2025**

Abstract

In the field of computed tomography, surface extraction from volumetric images is a fundamental capability with applications in both medical and industrial contexts. One particular application in the industrial field is metrology, the science of measurements, a discipline that imposes particularly strict accuracy requirements.

This thesis investigates surface extraction techniques, focusing on a pipeline that first estimates a rough surface and then refines it using sub-voxelic methods. We evaluated several algorithms from the open-source library *Visual Toolkit* by comparing the extracted surfaces to a reference model and assessing whether it satisfies some reasonable metrology criterion. These experiments were conducted on CT data acquired using both circular and helical scanning geometries to verify the hypothesis that more accurately reconstructed input data would yield more accurately extracted surfaces. While helical scans improved the quality of the extracted surface, the results were not accurate enough to satisfy the metrology criterion.

We then implemented and analyzed two sub-voxelic refinement techniques: a method based on the center of mass of the image values and a method based on the gradient of the image values. The latter showed greater potential, leading us to conduct a study of its parameters and design choices, including the method of gradient computation, interpolation methods, and the method used to estimate the orientation of the surface normal. With an optimized configuration and high quality of input data, our method produced surfaces where 60% of the points met the metrology criterion, and the mean error also remained below the specified threshold.

Although we did not manage to achieve full compliance, our results indicate promise for applications such as the detection of porosities or the ability to get a precise slice view of the object. We also got a better understanding of the effect of our design choices and determined that future research should be focused on accurately reconstructing sharp features.

Acknowledgements

I would like to express my sincere gratitude to everyone at X-RIS for their very warm welcome and for inviting me to several company outings, where I had great fun and got to meet lots of nice and interesting people. Also, I want to thank them for letting me use their demo room as a quasi-personal workspace and for giving me access to their development computers.

In particular, I want to thank Vincent, who co-supervised this master's thesis, for his great patience and for providing lots of support and explanations throughout the entire time I spent at X-RIS. His daily check-ups were very valuable, and I believe one could hardly find a more attentive mentor.

I also want to thank Professor Christophe Geuzaine, for his availability, as our weekly meetings truly helped structure the thesis, and for his insights, which helped me overcome some obstacles and keep progressing.

Finally, I want to thank my friends and family for their attentive ears and support.

Contents

Introduction

Context

This master's thesis takes place in the context of an internship at *X-RIS*, short for *X-Ray Imaging Solutions*, a Belgian company that specializes in the development of user-friendly digital X-ray imaging software as well as the design and manufacturing of digital X-ray devices. Founded in 2010 by Nicolas Poelst and Christophe Greffe, it is based in Herstal, in the province of Liège, Belgium. The company offers a wide range of services which include the conception of custom pieces of equipment, the installation of said equipment, software demonstrations and after-sales services such as technical maintenance. It also provides a wide range of products that go from portable X-ray sources and portable X-ray detectors to automated X-ray systems equipped with a turntable or robotic arm, allowing complex parts to be X-rayed from multiple angles. Moreover, they developed *Maestro*, a user-friendly visualization and automation software to process the X-ray images. This software consists of several modules enabling multiple features such as post-processing for image enhancement, automation of image acquisitions, 3D reconstruction, automatic defect recognition and storage management.

X-RIS frequently collaborates with other actors in the industry as well as academic partners. For instance, they collaborated on a joint project called *Automatic Defect Recognition in Industrial Control* with Euresys and Optrion, the university of Liège and the university of Louvain.

With significant international customers such as Safran, Total, Sonaca or the FBI, the solutions offered by X-RIS find their application in several fields such as the medical, industrial, and security sectors and even in the cultural sector. The purpose of these devices and software is to allow users to conduct non-destructive testing (NDT), a set of techniques that allows the construction of a digital representation of the studied object without altering its integrity. This proves to be particularly useful in the sectors aforementioned. For example, actors in the field of security may use this technology to safely inspect explosive devices or to help automate the detection of threats. In the industrial sector, NDT may be used to inspect manufactured pieces with the aim of detecting defects such as cracks or porosities in the structure of the object. In the cultural sector, NDT has been used for the conservation and observation of fragile antiquities and art pieces.

Computed Tomography

An important feature offered by X-RIS's maestro software is the ability to automatically perform computed tomography (CT) scans. A CT scan is an imaging technique performed with the goal of obtaining detailed internal images of a person or an object. These images can then be combined to generate a digital 3D representation of the subject. These scans are performed according to the following procedure:

1. The object to be studied is placed between an X-ray generating device and an X-ray detector device.
2. A X-ray shaped like a cone is generated and traverses the object as well as the air around it, which attenuates its intensity depending on the material's density and thickness. The intensity of the beam is measured at each pixel of the detector device, producing a grayscale image of the object's attenuation properties between the X-ray source and detector.

3. The object to scan is rotated by a small projection angle and a new X-ray shot is performed. This step is repeated until a sufficient number of projections has been acquired.

4. A tomographic reconstruction algorithm combines the different images acquired during the previous steps to obtain a 3D image where each pixel represents the density of the object's material and of the air around it. This is made possible thanks to the different view angles, which yield knowledge about the dimensions of the object, allowing the algorithm to take the thickness of the object into account when computing the attenuation coefficient of each pixel from the measured pixel's intensity.

The ability to obtain a 3D representation of an object, including its inside, allows for multiple use cases in the industrial domain such as flaw detection, assembly and failure analysis, or even reverse engineering.

One specific use case is the ability to measure an object's dimensions. This particular discipline of sciences is called metrology. Adding such a feature to the *Maestro* software would be particularly interesting for X-RIS as many of their partners would benefit greatly from it. Indeed, many industrial sectors and particularly the sector of aeronautics implement strict verification procedures concerning the dimensions of manufactured equipment. Figure 1 illustrates how metrology analysis can be performed on a digital representation of an industrial part.

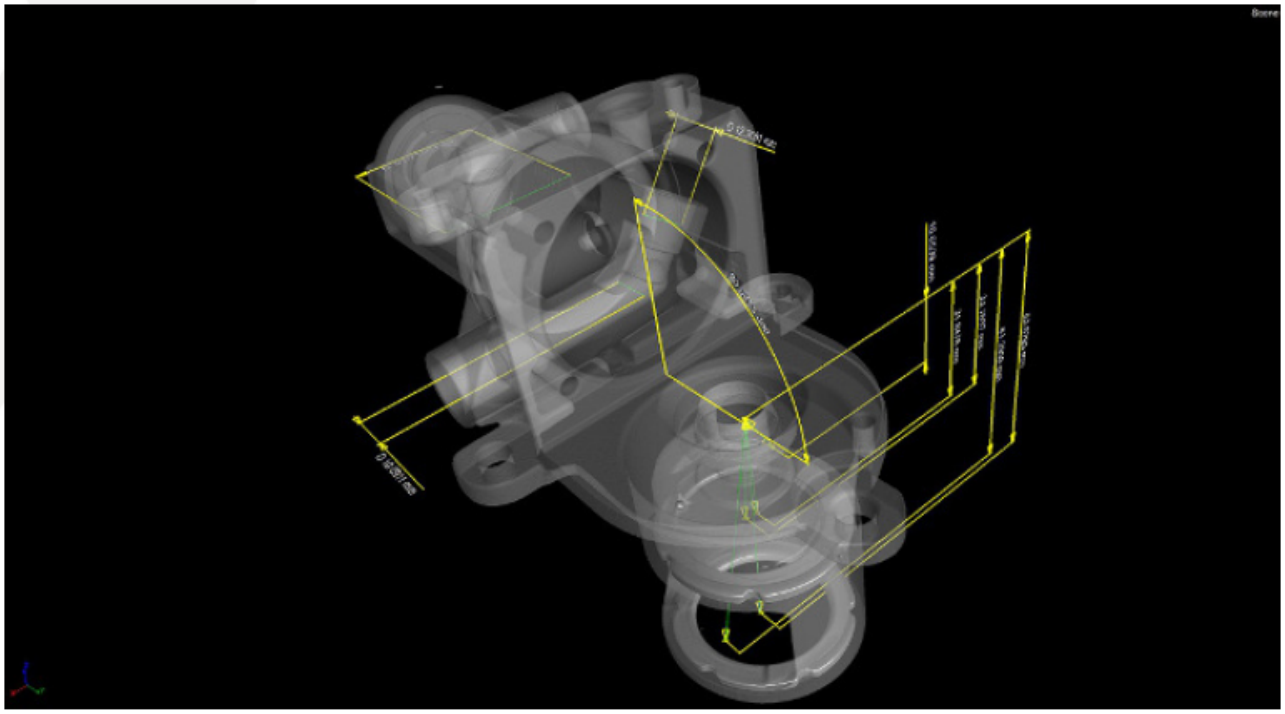


Figure 1: Illustration of surface extraction for metrology applications. Source : North Star Imaging, Inc. [1].

Nonetheless, the accuracy of the metrology analysis of an object is bounded by the quality of the reconstructed 3D model of the object. The metrology criterion used by *VGSTUDIO*, leader in CT reconstruction and visualization for industrial applications with nearly 80% of the market share, is to have a measurement uncertainty smaller than the tenth of the size of one voxel. To obtain such a level of accuracy, the 3D image obtained via CT scanning must be processed to extract a surface representing the interface between the object and the air around it. Optimally, this surface should be determined in such a way that the distance between a surface point and the closest point on a reference surface is always inferior to 10% of the size of one voxel. Higher levels of accuracy may even be required, depending on the client and their application. Moreover, an additional constraint is that the surface extraction process

should be achieved in reasonable time.

It is worth noting that surface extraction has a range of applications not limited to metrology. These applications typically require less precision on the reconstruction than for metrological applications. Some examples are :

- Detection of porosities inside the material. Indeed, as porosities have the same density as air, being able to differentiate the inside of the volume from the outside of the volume allows for the detection of porosities where air is detected inside the volume. Figure 2 shows an example of CT reconstruction for an aluminum object. Porosities (air bubbles) are clearly visible at different locations inside the object, but extracting a surface would permit automatic detection of this kind of production flaws.
- Ability to find a plane that is parallel to some side of the object, and use this plane to get a precise slice view of the object.

Figure 2: 3D image obtained via CT reconstruction of an aluminum part. Radiography allows the detection of porosities, which are visible at different locations inside the object without the need to alter the structure of the object. Source : X-RIS CT reconstruction.

As such, exploring potential solutions for surface determination from a 3D CT reconstruction constitutes the main objective of this master's thesis.

Obtaining a surface that would be deemed faithful to the original real life object is not a trivial process. Indeed, many artifacts will occur at every step of the reconstruction process and will alter the quality of the end result. For instance, some errors are inherent to the use of x-rays to digitalize the objects.

One such artifact is called the *beam hardening* phenomenon, which happens because the X-ray beam is made of polychromatic light. As such, its photons of lower energies may be totally absorbed by the material, which acts as a high-pass filter, effectively decreasing the average intensity measured by the detector. This decrease in intensity is interpreted by the X-ray detector as an increase in attenuation and therefore an increase in material density. In particular, the thicker sections of the material absorb more of these lower energy photons, leading to abnormally higher material density in these sections of the analyzed object. An example of this phenomenon is illustrated on Figure 3, where a CT scan is performed on a cylindrical object. Since photons traversing the center are traveling though a thicker section, the density appears higher towards the center of the object due to beam hardening.

There exist hardware and software solutions to help alleviate the effect of such artifacts. For example, a metallic filter can be used to pre-harden the beam by attenuating its lower energy photons before it even enters the object.

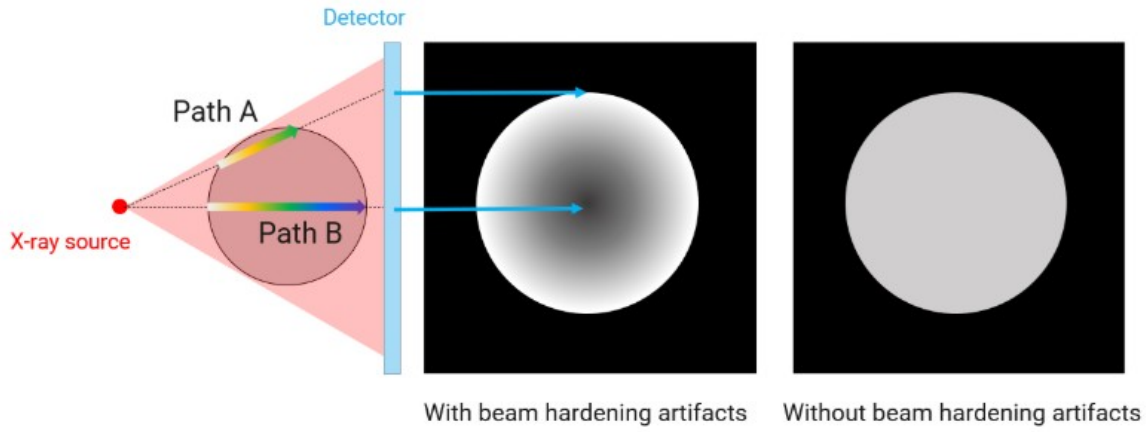


Figure 3: CT acquisition schematic setup with an illustration of the effect of the cylindrical object's section thickness on the beam hardening. Darker shades of gray correspond to higher material densities. Source : Rigaku Corporation [2]

Another type of artifact encountered during CT is the *cone beam* artifact. It is a consequence of the acquisition geometry employed by the CT scanner. Indeed, most CT scanners follow a circular path to acquire the different X-ray images, because it is the simplest mechanism to implement, as the object can simply be positioned on a turntable that rotates between each acquisition. However, sampling theory states that using a circular acquisition geometry does not provide enough information to obtain an exact reconstruction. Intuitively, it can be explained by the fact that the farther away from the plane in which the rotating source lies, the lower the sampling density becomes, as the X-ray beam is shaped like a cone. This lack of information causes a decrease in the quality of the reconstructed image.

To counteract this phenomenon, one solution is to use more complex trajectories when acquiring images. One such trajectory consists in continuously moving the object vertically while it rotates, generating a helical acquisition geometry. Another option consists in stopping the rotation at certain specific angles to move the object vertically, generating a "circle and vertical lines" trajectory. Figure 4 gives a visual representation of these two geometries.

To illustrate how these acquisition geometries can improve the quality of the reconstructed image, Figure 5 shows an example of cone beam artifacts on a Defrise phantom, and how using helical acquisition reduces these artifacts.

As these artifacts have an impact on the quality of the reconstructed 3D image, they also impact the quality of the surface extracted from the image.

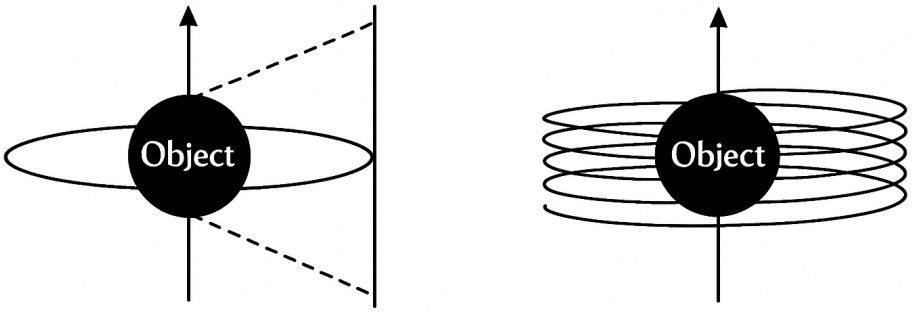


Figure 4: Schematic representation of some image acquisition geometries. Left : "circle and vertical lines" trajectory. The object rotates, following a circular path, but stops at certain angles to be moved vertically. Right : the object continuously moves vertically while in rotation, generating a helical geometry of acquisition.

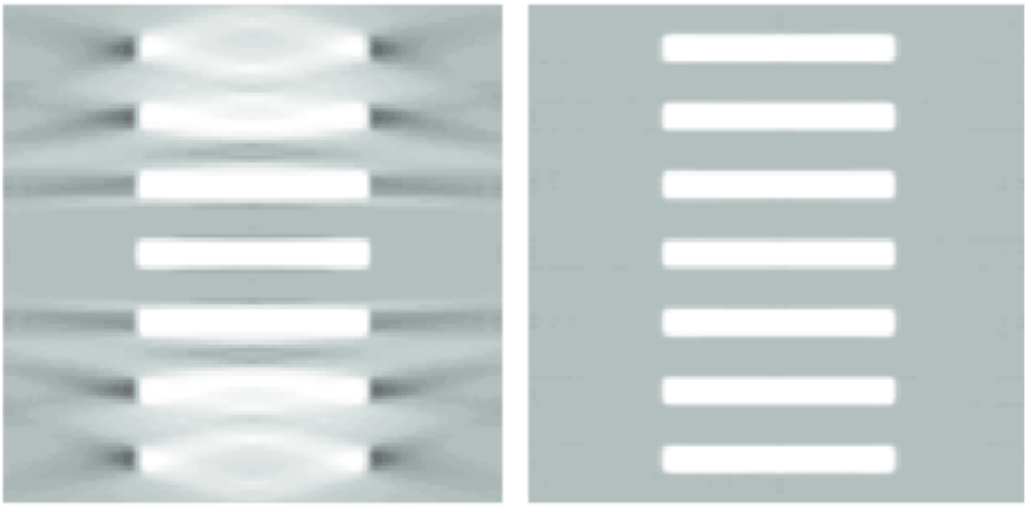


Figure 5: Left : CT scan of a Defrise phantom performed using circular acquisition geometry. Right : CT scan of a Defrise phantom performed using helical acquisition geometry. The impact of the cone beam artifact can be limited using helical instead of circular geometry. Reproduced from Taguchi et al., "Cone-beam image reconstruction using spherical harmonics" [3], 2001.

Objectives

The objective of this thesis is to evaluate and improve, if possible, the capabilities of the surface reconstruction technique based on the work of Roland Greffe, whose own master's thesis [4] was partly aimed at searching for a satisfactory and readily available surface extraction technique. In particular, we are looking for a solution capable of generating surfaces that satisfy the metrology criterion. To satisfy this criterion, the deviation between any point on the reconstructed surface and a reference model cannot exceed a tenth of a voxel. In other words, we want the surface to be accurately determined everywhere with an error of at most 10% of the size of a voxel. Moreover, the surface extraction process should be completed within a few tens of seconds at most.

Outline

The first step of this work consists of a study of the state of the art of surface extraction. We first describe the two main frameworks for surface extraction methods designed specifically for CT images. For one such framework, several sub-methods exist and are investigated. In particular, a description of every algorithm that had already been tested in R. Greffe's master's thesis is presented, along with their advantages and drawbacks. This study is presented in Chapter 1.

In Chapter 2, we perform the experimental study of the impact of the acquisition geometry on the quality of the extracted surface. In particular, we describe in detail the data used as input, the chosen software library, the pipeline and the evaluation methodology. Moreover, we investigate a correction technique aimed at improving those results.

In Chapter 3, we implement two promising methods we discovered during our research. These methods are both based on the idea of finding local voxel maxima of the image's gradient values, and further refine the positions of those maxima to obtain sub-voxel resolution. The two methods differ when refining the surface position, as the hypothesis motivating the first method is that the surface can be approximated by discontinuities in the gray values of the image, while the second method assumes that the surface can be approximated by discontinuities in the image's gradient values. The chapter concludes with a discussion of the results. To get an in-depth comprehension of the mechanisms that underpin the quality results, we perform a theoretical study on how the choice of parameters and methods affected the final reconstruction. We finish the chapter with a discussion about the aspects that remain unsatisfactory and where further effort should be focused.

Finally, a general conclusion is drawn, with a discussion about possible improvements.

Chapter 1

State of the art

The history of isosurface extraction for practical applications starts in 1987 with the publishing of the *Marching cubes* [5] algorithm, written by William E. Lorensen and Harvey E. Cline. The simplicity and speed of this algorithm made it the standard for isosurface extraction, even to this day as it is still relevant and widely used in practice [6]. However, this algorithm does not fit the requirements for all applications as it may, for instance, lack in accuracy. For this reason, many other techniques have since been described as an answer to the different objectives sought by isosurface extraction applications. More recently, we also see the emergence of new studies that seek to harness the power of artificial intelligence to come up with innovative solutions, as illustrated by the *FlexiCubes* [7] algorithm.

Some of these techniques are presented in the next sections, starting from the seminal *Marching cubes* algorithm. We first describe the algorithms that are already implemented in open-source software libraries, as our goal is to assess whether readily available methods yield satisfactory results. For each of these algorithms, we provide a short summary of its inner workings, as well as some of its advantages and drawbacks with regard to the needs of our application. Next, we report the results from the review article [8] that assesses how relevant some surface extraction techniques are specifically for metrology applications. Based on the conclusion drawn in [8], we present two promising sub-voxel refinement methods: one that is gradient-based and one based on the center of mass.

1.1 Marching Cubes Algorithm

The marching cubes algorithm's fundamental principle is to use a divide-and-conquer approach in order to locally assess the orientation of the surface. First, a threshold must be manually set in order to determine whether a voxel is inside or outside the volume when we look at its intensity value. After that a set of eight voxels, or eight cubes, will "march" across the entire volume. Each cube is assigned a label based on whether the underlying voxel is considered inside or outside the volume with regard to the threshold previously defined. Since there are eight cubes inside a group, each group is described by 1 of the 2^8 possible combinations of these cubes. A surface orientation can then be assigned to each combination, and will reflect the orientation of the surface, if any, enclosed between those eight cubes. The obtained combination can then be used as input in a lookup table for a fast determination of the surface orientation. Moreover, many of these orientations are symmetrical to others, and the lookup table can be reduced to 15 different entries, from the 256 initial ones, as represented on Figure1.1.

This method is very simple to understand and implement, and it is also very fast. The main drawback is that this simplicity comes at the cost of accuracy, since the surface defined locally by the eight cubes can only be one of 15 different ones, instead of the infinity of orientations that a natural surface may display. A first improvement to this is described in the original article and suggests performing a linear interpolation to precise the location of the vertices forming the surface. As the first and most standard algorithm for surface extraction, we expect to find other isosurface extraction algorithms that derive from

the marching cubes.

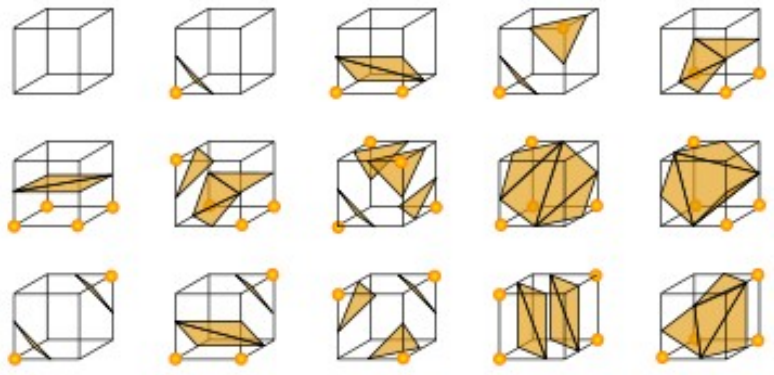


Figure 1.1: The 15 different surface orientations in the marching cubes method, as presented by Matthew Fisher (2014) on the Stanford University website [9].

1.2 Poisson Surface Reconstruction

Description

Invented by Kazhdan et al. [10], the Poisson surface reconstruction algorithm is an adaptation of the marching cube method.

The first step consists in determining an implicit function. In particular, an *indicator function* χ is defined whose value is 1 inside the model and 0 outside. Computing the gradient of this indicator function would yield a vector field with null vectors everywhere except at the border between the inside and outside of the model. When evaluated at these particular points, the gradient should yield vectors whose orientation is equal to the surface normal, except that they would be facing inward instead of outward. Therefore, the normals to the input point samples can be used as samples of the gradient of χ.

With this in mind, χ can now be found by solving:

$$\min_{\chi} ||\nabla\chi - \vec{V}||^2,$$

where $\vec{V}$ is the vector field obtained from the normals to the point samples. To find the best fitting values for χ in the least-squares sense, the divergence operator is applied to the previous equation, which becomes:

$$\nabla \cdot \nabla\chi = \nabla \cdot \vec{V}$$

which is a standard Poisson problem.

To avoid unbounded values being computed for the gradient, the indicator function is convolved with a Gaussian filter whose variance is of the order of the sampling resolution.

To obtain a sufficiently precise space resolution around the sample points, an adaptive octree is used. Once an estimation $\tilde{\chi}$ of the indicator function is found, it is evaluated at the sampled points and averaged to use as an isovalue for the isosurface extraction step, where a high-resolution adaptation of the marching cubes algorithm for octrees is used. This adaptation consists in ensuring consistency across the different octree depths to avoid cracks.

Advantages

- Poisson systems have been the objects of numerous inter-disciplinary research studies that have developed many robust and efficient Poisson solvers.

- Using adaptive Poisson solvers in the vicinity of the surface allows for the implementation of a reconstruction algorithm whose spatial and temporal complexities are proportional to the size of the reconstructed surface, thanks to the use of octrees.
- The algorithm estimates the local sample density to generate a reconstruction that maintains sharp features in areas of dense sampling and provides a smooth fit in sparsely sampled regions.
- The reconstructed surface is guaranteed to be watertight.
- Solving the Poisson system for the entire set of input points is a global solution, in the sense that all surface points contribute to the solution, which helps in reducing local inconsistencies.
- Using octrees is the key element for providing the high spatial scalability of this technique.

Disadvantages

- The quality of the reconstruction depends on the estimation of the point normals, which is sensitive to noise.
- As we tend to have noisy positions and normal estimates around sharp edges, this algorithm adapts both the scale and the variance of the samples in order to obtain a smoother reconstruction, without sacrificing fidelity in areas of dense sampling. This can be viewed as an advantage, but as the sampling around edges tends to not be very dense, the edges may end up too smoothed.
- The algorithm may connect independent regions due to the lack of sampling between them.
- There is no guarantee that the points constituting the reconstructed surface will match the original point samples, which could be unacceptable depending on the desired guarantees for the final application.

1.3 Extract Surface From Range Images

Description

Proposed by Levoy et al. in their 1992 paper "Extracting Surfaces from Gradually-Varying Density Volumes" [11], this surface extraction algorithm is also based on the marching cube method and can be described by the following steps :

1. The input is a set of range images. In 3D, these images are typically acquired in the following way : a laser stripe spreads a laser beam onto the object, and the reflection of the beam is captured by an observing sensor, from which a depth profile is computed and stored as a range image. The object is then rotated to obtain several range images. A 3D voxel grid is used to integrate the data from the range images, where each voxel contains two fields :
   - The value of the Signed Distance Function $D(X)$, which represents the distance to the nearest surface point in the sensor's line of sight.
   - The value of the Weight Function $W(X)$, which is a measure of the reliability of the range image. For example, images taken from grazing angles or near the noisier mesh edges will be assigned lower weight. These functions are updated cumulatively by averaging their values from all range images in the following way :

$$D_{i+1}(X) = \frac{W_i(X)D_i(X) + w_{i+1}(X)d_{i+1}(X)}{W_i(X) + w_{i+1}(X)},$$

$$W_{i+1}(X) = W_i(X) + w_{i+1}(X),$$

where i represents the ith range image. Based on the value of the Signed Distance Function $D(X)$, all the voxels that are in front of the observed surface are labeled as "*empty*" and can be discarded during further processing to improve performance, and the voxels behind the surface or never observed (meaning that the associated weight is 0) are labeled as "*unseen*".

2. The marching cube algorithm is now applied to extract an isosurface at the zero-crossing of the Signed Distance Function $D(X)$.

3. Direct transitions between "*empty*" and "*unseen*" voxels are triangulated to create a watertight surface, in what is called the "*hole filling*" step.

Advantages

- The method is robust to noise as several range images from different angles are integrated together and a weight function is used to take into account the reliability of the range data.
- The space optimization enabled by the voxel labeling allows for high scalability, as demonstrated in the results section of the paper, where 70 range images are combined to create a surface consisting of 2.6 million triangle cells.
- The ability to fill holes and obtain a watertight surface.
- For a voxel of size 0.35mm, the results section of the paper shows a root mean square distance (RMS) of 0.1 mm, which is sub-voxelic precision, even though not under the metrology precision tolerance.

Disadvantages

- The quality of reconstruction for thin objects can be worse as the Signed Distance Functions from opposite sides could interfere with one another.
- It requires the appropriate setup to capture the range images as input. As such, CT scans do not yield the appropriate type of input data.
- This method does not allow for the reconstruction of internal surfaces as the laser beam do not traverse the object. This is a limiting factor if the surface is meant to be used to detect internal imperfections such as air bubbles.

1.4 Surface Reconstruction From Unorganized Points

Description

The paper titled "Surface Reconstruction from Unorganized Points" [12], written by Hoppe et al., describes a method for surface reconstruction from a point cloud without prior knowledge about the topology or the geometry of the object. Similarly to the method described in Section 1.3, this method relies on the computation of a Signed Distance Function, from which the zero-crossings will be used to extract an isosurface using the marching cubes method. Contrarily to the previous method, the input data is a set of unorganized points $X = \{x_1, ..., x_n\} \subset \mathbb{R}^3$, therefore the method used to compute the Signed Distance Function will be different. The algorithm can be described in the following steps :

1. To compute the Signed Distance Function, an oriented plane is associated with each point x_i. These tangent planes $T_p(x_i)$ are local linear approximations of the surface to reconstruct, but their union is not used to directly define the output surface as it can be incoherent and non-manifold. A tangent plane $T_p(x_i)$ is defined by a point o_i called the "*center*", and a normal vector $\hat{n}_i$. To compute those two properties, a neighborhood of points $Nbhd(x_i)$ centered around x_i is used. o_i is computed as the centroid of $Nbhd(x_i)$, and $\hat{n}_i$ is determined using principal component analysis :

   (a) Compute the 3x3 covariance matrix $CV_i = \sum_{y \in Nbhd(x_i)} (y - o_i) \otimes (y - o_i)$, where $\otimes$ denotes the outer product vector operator.

   (b) The eigenvalues $\lambda_i^1 \geq \lambda_i^2 \geq \lambda_i^3$ of CV_i are computed. $\hat{v}_i^3$, the eigenvector associated with λ_i^3, is the direction of least variance since λ_i^3 is the smallest eigenvalue.

   (c) To ensure consistent normal orientation, either $\hat{v}_i^3$ or $-\hat{v}_i^3$ will be chosen. To know which one to choose, a graph based on the principle that close points in space share a similar orientation is constructed in the following way :

      i. The nodes of this graph represent the centers of the tangent planes and the edges connect planes whose centers are close.

      ii. An Euclidean Minimum Spanning Tree (EMST) is used as it is simple and connected, and is relevant for representing neighbor proximity. An example of such EMST is visible on Figure 1.2 C).

      iii. As the EMST is still not sufficiently dense in edges, more edges are added using the following strategy : and edge (i, j) is added if o_i is in the k-neighborhood of o_j or vice-versa. Figure 1.2 D) shows an example of graph resulting from the increase in edge density.

      iv. To ensure an optimal order of traversal of the graph to propagate the normal orientation, a cost $1 - |\hat{n}_i \cdot \hat{n}_j|$ is added to each edge (i, j), and a minimal spanning tree of the resulting graph is computed. This is advantageous because it tends to propagate the orientation along the directions of low curvatures first, avoiding the ambiguous situations encountered around sharp edges. An example of traversal order for orientation propagation is represented on Figure 1.2 E).

      v. Normal orientations are finally chosen in the following way : if $T_p(x_i)$ has been assigned orientation $\hat{n}_i$ and $T_p(x_j)$ is the next plane in the graph, then $\hat{n}_j$ becomes $-\hat{n}_j$ if $\hat{n}_i \cdot \hat{n}_j < 0$. Figure 1.2 F) shows a set of tangent planes with the computed orientation.

   The Signed Distance Function can now be computed according to the following principle : the distance from a point $p \in \mathbb{R}^3$ to the surface to reconstruct can be approximated by projecting p onto the tangent plane $T_p(x_i)$ associated with the center point o_i which is the closest to p. The position of the projected point z is therefore computed as

$$z = o_i - ((p - o_i) \cdot \hat{n}_i)\hat{n}_i,$$

   and the signed distance is computed as :

$$f(p) = (p - o_i) \cdot \hat{n}_i.$$

   Computing the new project position z is useful because if the distance $d(z, X)$ between z and the closest sample point is larger than a certain value based on the local sample point density and noise, p is considered outside of the boundary of the object and $f(p)$ is undefined.

2. The zero-crossing of the Signed Distance Function is used by the marching cubes method to extract an isosurface, while discarding the cubes where the Signed Distance Function was marked undefined at the previous steps.

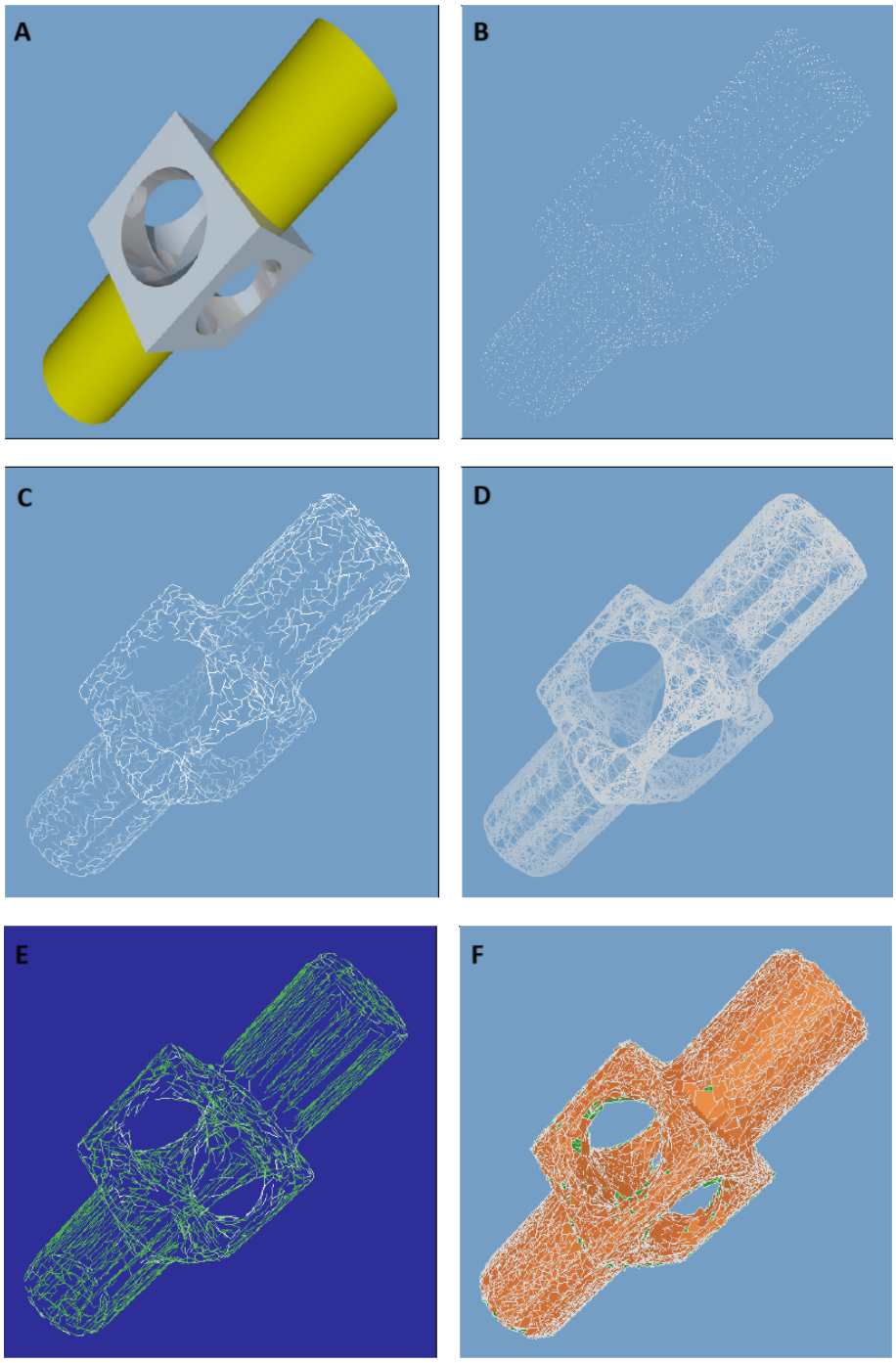


Figure 1.2: Steps involved in Hoppe's surface reconstruction algorithm for the computation of normal orientations. A) 3D object from which a point cloud is extracted. B) Point cloud representing the set of unorganized points $X = \{x_1, ..., x_n\}$ used as input for the reconstruction algorithm. C) An EMST where each node is the centroid of a point's neighborhood. Centroids that are close enough are linked with an edge. D) The EMST is made denser by adding more edges between two nodes whose neighborhoods overlap. E) A Minimum Spanning Tree is computed, from which a traversal order for orientation propagation is deduced. F) Representation of the tangent planes oriented according to the orientation propagation policy, which are used to compute the Signed Distance Function.
Reproduced from Hoppe et al., "Surface Reconstruction from Unorganized Points", 1992.

Advantages

- The output surface is guaranteed to be made of triangular cells only.
- Boundaries are handled where the Signed Distance Function is undefined.
- The neighborhood size parameter can be tuned. Higher values increase the quality of the reconstruction when the spread of points is uneven, but also lead to a reduction in quality around sharp edges.

Disadvantages

- The initial unorganized point cloud is not included in the final extracted surface, which may be unacceptable depending on the application.
- The preprocessing step of computing the tangent planes before applying the marching cubes involves the creation and traversal of graphs, which can be computationally intensive depending on the number of neighbors chosen. The results section shows that for a femur consisting of 18224 points, acquired via CT, with a neighborhood size of 40, the reconstruction takes 2135 seconds. However, we can expect it to run faster since the article was published in 1992, and computers were then much less powerful.

1.5 Flying Edges

Description

Introduced by Schroeder et al. in their paper titled "Flying Edges: A High-Performance Scalable Isocontouring Algorithm" [13], this method aims at improving the scalability of the marching cubes technique by leveraging the parallelization abilities of modern systems. To this end, several steps must be performed in order :

1. Since isocountouring algorithms produce an indeterminate number of output points and cells, dynamic memory reallocation is usually required throughout such algorithms. The first step of the flying edges method consists in parsing the data to know in advance the output size. Therefore, there is no need for some costly dynamic reallocation and the output memory can be easily partitioned between the different threads. This step consists of two passes over the data, where each row is processed in parallel, independently of each other :
   (a) During the first pass, each edge in the x direction between two voxels is checked to determine if it intersects the isocontour, assuming x is the fastest varying data direction, to take advantage of cache locality. This yields the positions of the leftmost and rightmost intersections of the isocontour with the edges between voxels in the x direction for each row, allowing the "trimming" of voxels located left and right of these boundaries, since we know that the isocontour does not traverse those voxels. This trimming operations may allow to skip entire rows of voxels.
   (b) During the second pass, only the voxels inside the trimmed window are processed for each row. For each voxel, a table lookup is performed to determine the corresponding marching cube surface and whether the y- and z-cell intersect the isocontour. The number of output points and triangles for each row can now be deduced from this information.

   An 2D example is shown on Figure 1.3 to illustrate the resulting metadata after the first step.

2. The second step consists in integrating the information about the number of output points and triangles of each row, to be able to allocate the total amount of output memory. This step can also be parallelized to increase the scalability.

3. During the final step of the algorithm, each row of voxels (inside the trimmed window) is processed in parallel to compute the intersection points positions and the construction of the triangle cells, following the principles of the marching cubes method.

Parallelization occurs at every step of the process, allowing for high scalability efficiency.

Advantages

- The main advantage of this technique is that it fully leverages the capabilities of modern computers. It is highly scalable as shown in the results section of the paper. For example, the authors show that a reconstruction made using 36 threads took 8.58 s when the regular marching cubes method took 69.86 s.
- The algorithm is able to process data larger than the size of the GPU memory.
- The input data is a 3D image with gray level values, as is the output of a CT reconstruction.
- Contrarily to other methods, the flying edges algorithm does not need to preprocess the input data to build a rapid search structure (such as an octree). These structures are memory-consuming and the preprocessing step can take a significant amount of time.
- Since 4 voxels share an edge, many marching cubes methods visit a same edge up to four times. These algorithms often use locators to merge the resulting coincident points together. The design of the flying edge method is such that it avoids visiting an edge more than once, removing the need for data structures to merge the coincident points entirely.

Disadvantages

- This algorithm suffers from the same limitations as the original marching cubes method in terms of quality of the reconstructed surface, as it only improves the scalability of the technique.

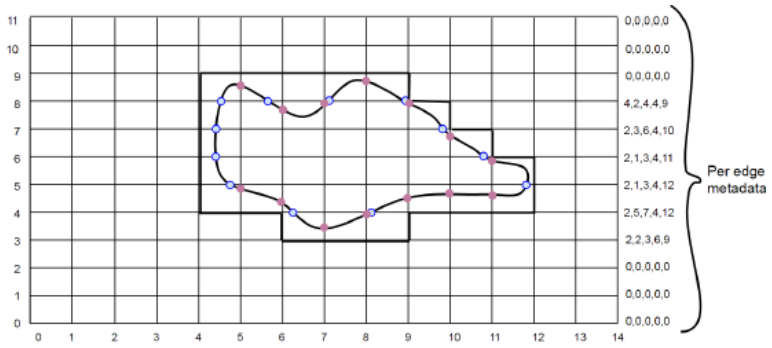


Figure 1.3: A 2d illustration of the isocontour and the trimmed voxels. On the right, the metadata generated after the first two passes on the data consists of the number of x- and y-edge intersections with the contour, the number of output triangles for the row, and the positions of the beginning and end of the trimming window.
Reproduced from Schroeder et al., "Flying Edges: A High-Performance Scalable Isocontouring Algorithm", 2015.

1.6 Surface Nets

Description

The Surface Nets method, described by Schroeder et al. in the paper title "A High-Performance SurfaceNets Discrete Isocontouring Algorithm" [14], is an isocontouring method also designed to leverage the parallelization capabilities of modern computer systems. As such, it shares a lot of similarities with the Flying Edges method described in Section 1.5. Another driver for the design of this method is the ability to contour discrete scalar fields, such as label maps, where the original marching cubes method is designed to contour continuous scalar fields, in the sense that the scalar value is assumed to vary linearly across an edge. This algorithm also differs from the flying edges method in the way it generates the surface points. Indeed, the marching cubes algorithm generates one point at every intersection between an edge and the isocontour, while the Surface Nets method generates at most one point per voxel if at least one edge is intersected. The intersections of the isocontour with each voxel are then used to define a smoothing stencil consisting of up to six edges connecting voxel face neighbors which also contain a generated point. A Laplacian smoothing process is then performed using those stencils, constraining the points within the voxel from which they originate, while moving those points along boundaries surfaces. Finally, a triangulation filtering process is applied to the output as the smoothing operation generates quadrilateral polygons.

These operations can be described by the following succession of steps :

1. The first step consists in passing twice over the data to determine the number of points and triangles in the final surface, allowing the allocation of output memory to be partitioned between the different threads.

   (a) Rows of voxels are parsed in parallel in the x direction, assuming x is the fastest varying data direction to take advantage of cache locality, in order to detect which voxel edges cross the contour in the x direction and to trim empty voxels. This phase is exactly equal to the first phase of the Flying Edges algorithm, with the exception that an additional field is stored for each voxel : the label region in which the voxel belongs.

   (b) The purpose of the second pass is to check for y- and z-edge intersections, exactly like described in the flying edges method. However, the label information gathered during the first pass allows further optimization of time by skipping edge intersection checks if both endpoints the edge have the same label. The trim interval that was determined during the first pass can now be shortened further.

   At the end of this step, metadata about the number of points and primitives is associated to each row of voxels.

2. The metadata of each row is now integrated concurrently to obtain the total number of points and primitives constituting the output, which allows for precise output memory allocation.

3. Using information from the previous phases, each row of voxels is processed in parallel to produce center points and stencils. These stencils are stored as pairs of point indices and used in the Laplacian smoothing phase, as they represent the connection between neighboring voxels sharing a face. Moreover, quadrilateral polygons connecting the center points of voxels that share an edge are created. These quadrilaterals are a first approximation of the isosurface but still need to be processed by the Laplacian smoothing filter.

4. The next steps consists in smoothing the output of the previous step using an iterative Laplacian smoothing filter, which works in the following way : each point p_i is iteratively moved toward the center point Δp_j of its N connected points, and these connected points can be retrieved thanks to

the stencil associated to p_i. A convergence factor λ controls the amount of motion per iteration, given the following iterative equation :

$$p_i = p_i + \lambda \Delta p_j \text{ with } \Delta p_j = \frac{1}{N} \sum_{j=0}^{N} (p_j - p_i)$$

The authors choose to set the number of iterations to 25 but do not give precisions about the convergence factor's value, only that they keep it modest.

5. The final step consists in triangulating the quadrilateral polygons resulting from the previous phases. The authors mention that the choice of the triangulation method has little impact on the performance and appearance of the end result.

In terms of time performance, the authors show that the surface nets method strongly outperforms all the methods they compared it against, even the flying edges for a same amount of threads. Unfortunately, no comparison on the quality of the reconstructed surface is displayed.

Advantages

- It leverages the information of label maps from the input data to further trim the voxels than the flying edges method did. The results section of the paper shows that this algorithm is the fastest among all those tested, and that it displays high scalability efficiency.
- It preserves sharp features better than the marching cubes method and its variants, but sharp edges can still be averaged out during the Laplacian smoothing process.
- It does not share the same limitations in terms of surface approximations as the marching cubes methods.

Disadvantages

- It is designed to work with discrete label maps, meaning that a pre-processing phase to distinguish the object from the background is necessary if we want to work with CT reconstruction images as input.

1.7 PowerCrust Algorithm

Description

The powercrust algorithm was presented in 2001 by Amenta et al., in their research paper *"The Power Crust"* [15]. Their method consists in using the sample points to compute the *medial axis transform*, or MAT, of the object to reconstruct, and then use this MAT to generate a piecewise-linear surface.

The MAT can be seen as the "skeleton" of the object, and is a way of representing the signed distance function on the entire space, where the signed distance function is the function that returns the distance to the closest point on the surface. Intuitively, the inner MAT represents the infinite union of its maximal internal balls, and an example can be seen on Figure 1.4, but an outer MAT also exists and consists of the infinite union of the maximal external balls.

Among the existing techniques to approximate the MAT, the authors chose to compute the Voronoï diagram of the sample set. In such a diagram, a cell is associated to each sample s, such that any point inside the cell is closer in distance to s than to any other sample, as can be seen on Figure 1.4. With a sufficiently dense sampling density, the cells around every sample s should all be long, skinny and perpendicular to the surface. If this is the case, the opposite vertices of these long cells should lie near

the inner and outer medial axes of the object. This motivated the authors decision to make use of these Voronoï vertices as an approximation of the medial axes.

We can now reconstruct a set of balls centered on the Voronoï vertices, with a radius equal to the distance between the vertex and the closest sample point on the surface. These balls are called the *polar balls*. The next step consists in computing the power diagram of the obtained polar balls. In such a diagram, the power distance from each point in space to each of the polar balls is computed, and to every polar ball is associated a cell containing the points that minimize the power distance between those points and the ball compared to other balls.

Once the power diagram is obtained, the subsequent step is to label the poles as being inside or outside the object. This decision is based on the result formulated in the companion paper [16] : two adjacent polar balls associated with two outer poles or two inner poles intersect deeply, while the polar ball of an inner pole must intersect the polar ball of an outer pole shallowly. This level of intersection can be characterized by an angle α, that we can set as a parameter of the algorithm.

The labeling algorithm can be described by the following procedure:

1. Add eight artificial poles around the object to create a bounding box.
2. Label the poles adjacent to this bounding box as outer poles.
3. Propagate this labeling along the other poles following a greedy policy:
    - Any unlabeled pole whose polar ball intersects deeply with the previous one is also labeled as an outer pole.
    - Any pole whose polar ball intersects shallowly is labeled as an inner pole.
    - Propagate first from the poles in which we have the most confidence first, as a measure of robustness against noise.

We are now able to reconstruct the *powercrust*, which is the set of boundaries between the power diagram cells of inner and outer poles. These boundaries are two-dimensional shapes, as expected.

Advantages

- Any input yields an output surface that is watertight, which removes the need for related post-processing steps.
- Sensitivity to noise can be assessed, leading to the elimination of points on the medial axis if they are deemed unusable. If the angle between the vectors connecting the point on the medial axis to its two closest points on the surface is greater than a certain threshold, the point is discarded.
- The authors have found a method for improving the results around sharp corners. Indeed, the sample points around sharp corners induce inner Voronoï poles that do not pass the "skinnyness" test, while the outer poles will pass it. Following the previously described policy, these inner poles would be discarded, which would result in rounded corners in the final reconstruction. This behavior is not desirable, and the proposed solution is to let the user indicate whether the model contains sharp corners, in which case both the inner and outer poles will be discarded. This allows the cells corresponding to adjacent polar balls to extend where the previous poles where and to meet at a sharp corner. An example of this is shown on Figure 1.5.

Disadvantages

- The medial axis is very sensitive to noise. Indeed, small perturbations on the surface induce large modifications of the medial axis. This makes this algorithm not very suited for applications such as shape decomposition or feature recognition.

- The power shape defined for this method is based on the theory of α-shapes, which suffers from the difficulty of finding a good α to balance hole-filling and loss of detail when sampling is non-uniform.
- The mesh generated by this algorithm is made of polygonal cells, which imply that additional steps may be required if we want to work with triangular cells only.
- As the reconstructed surface is a union of boundaries between intersecting inner and outer power cells, there is no guarantee that it perfectly includes the input point cloud.

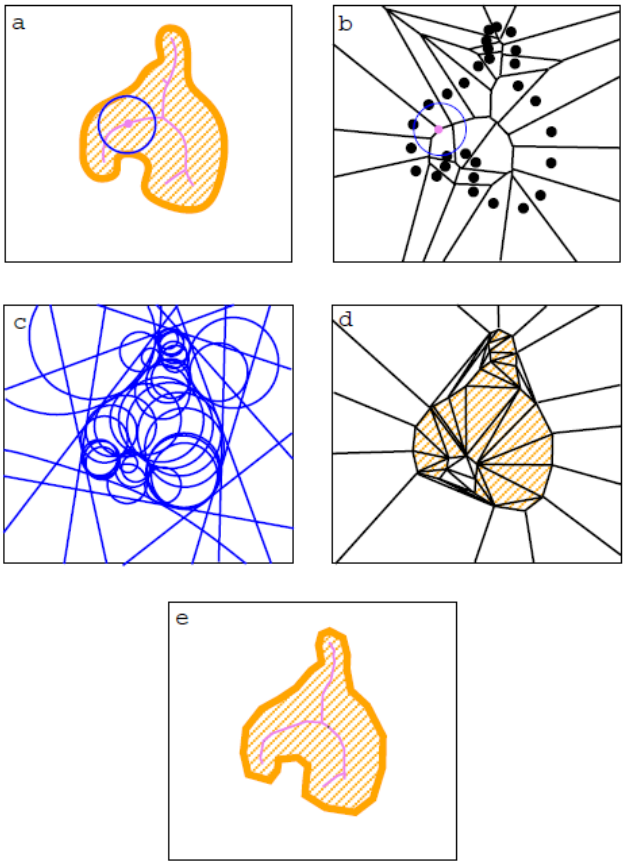


Figure 1.4: Steps involved in the PowerCrust algorithm for the reconstruction of the surface of a 2D object. a) The shape of the 2D object to reconstruct, with its true medial axis and an example of one maximal internal ball. b) From an input point cloud sampled on the surface, the Voronoi diagram is computed, with an example of a maximal internal ball. c) Polar balls are computed from the Voronoi vertices, with inner polar balls inside the point cloud and outer polar balls outside. Some of the outer polar balls centers are located at infinity, which explains why we obtain what looks like straight lines. d) The power diagram is now computed over the entire space, with each cell labeled inner or outer depending on the intersection angle between the polar balls. e) The final surface is the union of all interfaces between the inner and outer cells of the power diagram. Reproduced from Amenta et al., "The Power Crust", 2001.

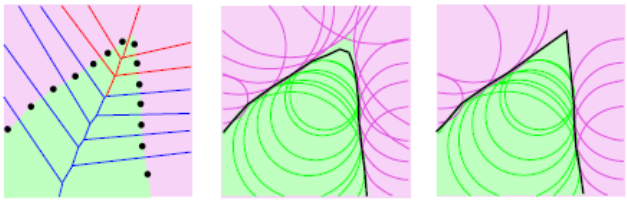


Figure 1.5: The "skinnyness" of the cells of the Voronoi diagram can be used to detect sharp corners. The quality of the reconstruction is improved around such corners by removing both the inner and outer polar balls associated with these Voronoi cells, and then extending the power diagram boundaries from nearby well-sampled regions until they meet in a sharp angle. Reproduced from Amenta et al., "The Power Crust", 2001.

1.8 Comparison of Methods Relevant for Metrology Applications

The article titled "*comparative of 3D surface extraction methods for potential metrology applications*" [8], written by Ontiveros-Zepeda et al., offers an overview of the most common methods for surface extraction, while also comparing them against each other to determine the methods that display the most potential for metrology applications.

Each extraction method is first succinctly described :

- **Atlas Guided** When the information from the gray level values of an image is not sufficient, manual segmentation can be performed to construct a spatial map called an atlas. This atlas must be aligned with the target image, and is used as a reference frame to extract the contour with more accuracy. The main limitation of this technique is that it requires manual intervention and that it is sensitive to the initial alignment with the target image.
- **Deformable Models** In this method the boundaries of the object are represented by parametric curves such as splines, and their shape will be influenced by internal and external forces. The main limitation is that the method relies heavily on the manual choice of a physical model and parameters.
- **Detection Of Discontinuities** In this approach, an edge is determined to be located at a discontinuity in the image's intensity, or at a discontinuity in the first or second order derivative of the intensity. The first step of this approach is to perform a voxel-level surface extraction, and the second step is to increase the resolution of the surface's position using sub-voxel refinement. The advantage of this method is that it is automatic, and has been widely studied for medical applications.
- **Pattern Classification** The goal of this kind of methods is to distinguish features from an image by learning from training data. One disadvantage of using such classifiers is that they do not perform spatial modeling an dare sensitive to noise.
- **Region Oriented** This method can be viewed as the process of partitioning the image into sub-regions. For example, seed points are chosen and regions grow around them by appending neighboring pixels who share similar properties, such gray value or edge sharpness. The main limitations are that those seed points must be chosen manually and are very sensitive to noise, which explains why this method is not commonly found in metrology applications.
- **Threshold Based** This simple method creates a binary partition of the image based on each pixel's intensity compared to a threshold value. This method is automatic, widely studied and has a wide range of implementations, but only in the medical field where the metrology criterion is less strict than in the industrial field.
- **Trainable Segmentation**This method consists in training artificial neural networks at performing image segmentation. Some limitations are the computational cost and the need for training data.

- **Watershed Transformation**When applied to an image, this transformation links together components of similar gray level in order to detect and label objects in the image. The process is the following : the grayscale image can be interpreted as a 3D landscape where the elevation represents the gray value, and the gradient of the gray values is represented as mountain ridges. The regional minima of the images are the points of lowest elevations, where water would collect from the elevations. As water fills this landscape, water basins start filling the regional minima. The edges where different basins meet are called the watershed lines and determine the image segmentation. A limitation is that the method often leads to over-segmentation in the presence of noise and other irregularities in the gradient.

Based on this analysis, the authors conclude that the *Detection of Discontinuities* and *Threshold Based* methods are the most promising for metrology applications. In particular, the local thresholding method is attractive because studies have shown that it yields accurate results while being very simple in concept. Among the methods based on discontinuities, they suggest using the Canny [17] algorithm as it provides good surface location capabilities and significantly reduces the data quantity by filtering non-useful information.

1.9 Canny Edge Detection With Center Of Mass

Description

The work described in Section 1.8 suggested promising results when using methods based on the discontinuity of the image's gray values, instead of methods based on a threshold gray value. Following these results, the article named "*A 3D edge detection technique for surface extraction in computed tomography for dimensional metrology applications*" [8], written by Yague-Fabra et al., describes such a technique. The first step of their algorithm consists in applying the Canny algorithm [17] to the input image in order to extract a first approximation of the surface. The Canny algorithm itself consists of four distinct steps :

1. A gaussian filter is applied to the input image, by convoluting it with a gaussian kernel to smooth out the noise.

2. Compute the gradients of the filtered image in the x, y and z directions in order to obtain the gradient magnitude for each voxel of the image.

3. Apply local non-maximum suppression in the direction of the gradient to ensure that the edge is only one voxel thick in the direction of the gradient and centered on the maximum gradient's magnitude value.

4. Two threshold values are chosen as parameters of the algorithm. If the gradient's magnitude of a voxel is higher than the first threshold, the voxel is considered part of the edge, and is called a strong edge. Otherwise, it can still be considered part of the edge if its gradient's magnitude value is higher than the second threshold and that the corresponding voxel is located next to a strong edge. It is then called a weak edge. This step is called *double thresholding*.

The image obtained after filtering the input image with Canny is already an estimation of the object's surface, but it is only precise up to one voxel in size, as an edge as defined by Canny is always one voxel thick, which is not sufficient for dimensional metrology applications. The idea of the paper is to extend the Canny technique by refining the position of the surface inside the edge voxel. To this end, the center of mass of the gray values of the voxels surrounding each edge voxel is computed in the following way :

$$X' = \frac{\sum_{i=1}^{i=3}(X_i \cdot G_{X,i})}{\sum_{i=1}^{i=3} G_{X,i}}, Y' = \frac{\sum_{j=1}^{j=3}(Y_j \cdot G_{Y,j})}{\sum_{j=1}^{j=3} G_{Y,j}}, Z' = \frac{\sum_{k=1}^{k=3}(Z_k \cdot G_{Z,k})}{\sum_{k=1}^{k=3} G_{Z,k}},$$

where X_i, Y_j and Z_k are the coordinates of the centers of the voxels, while $G_{X,i}$, $G_{Y,j}$ and $G_{Z,k}$ are the gray value associated to the voxels in the search window. The authors find that choosing a window of size 3 in every direction yields the optimal results for the sub-voxel resolution refinement, which explains the boundary values chosen for i, j and k in the previous equations.

When comparing this method to a simple thresholding method, the authors manage to obtain an improvement in precision of up to 60% in some cases.

Advantages

- Results show that this method handles multi-material transitions robustly, while the simple thresholding method fails.
- Results show that the method is able to achieve sub-voxel resolution of up to 0.01 voxel in precision.
- The window size for the computation of the center of mass can be optimized.

Disadvantages

- As a Gaussian filter is first applied to the image to attenuate the noise as part of the Canny algorithm, the edges of the surface to reconstruct will be smoothed out as well, which affects the quality of the reconstruction.
- Computing the gradient value in all three main directions may be computationally expensive in comparison to some of the simpler methods described in the previous sections.

1.10 Gradient-Based Surface Extraction Methods Comparison

Description

The work presented in the previous sections (1.8 and 1.9) demonstrated the potential of gradient based methods for surface extraction for dimensional metrology applications. The article named "*Analysis of Surface Extraction Methods Based on Gradient Operators for Computed Tomography in Metrology Applications*" [18] describes two such techniques and compares them.

The first technique is similar to the one we described in Section 1.9, based on the Canny algorithm followed by a sub-voxel refinement procedure. The difference lies in the sub-voxel resolution refinement technique, which is now based on the assumption that the surface is located where the discontinuity in the image's gradient is maximal, contrarily to the technique presented in section 1.9 where the gray values were used directly.

Similarly to the technique described in Section 1.9, the second method presented in this article consists in applying a gradient-based edge detection filter to the input image, before applying a sub-voxel resolution technique to refine the position of the surface inside these edge voxels. To perform the edge detection step, the Deriche operator is used recursively to compute the gradient values in each direction:

$$\gamma^+_{ijk} = X_{ijk-1} + 2e^{-\alpha}\gamma^+_{ijk-1} - e^{-2\alpha}\gamma^+_{ijk-2},$$

$$\gamma^-_{ijk} = -X_{ijk+1} + 2e^{-\alpha}\gamma^-_{ijk+1} - e^{-2\alpha}\gamma^-_{ijk+2},$$

$$\theta_{ijk} = -(1 - e^{-\alpha})^2(\gamma^+_{ijk} + \gamma^-_{ijk}),$$

where X_{ijk} is the gray value for the voxel ijk. The parameter α allows a trade-off between edge localization and noise suppression, and its optimal value is experimentally determined by the authors to be equal to 3.

The values of the gradients in each direction can then be combined to obtain the gradient's magnitude in voxel ijk using the following equation :

$$\theta_{ijk}^{XYZ} = |\theta_{ijk}^{X}| + |\theta_{ijk}^{Y}| + |\theta_{ijk}^{Z}| \tag{1.1}$$

Similarly to the Canny filter, the edge is now determined to be where the gradient's magnitude is maximal, following a local non-maximum suppression operation, but this edge is still one voxel thick.

The sub-voxel refinement technique described for this method consists in computing the gradient value in each of the 13 main directions surrounding each edge voxel, and to choose the direction that maximizes this gradient value as an approximation of the surface normal. The surface position will therefore be refined in the direction of this surface normal approximation. The new sub-voxelic position of the surface is computed to be the center of mass of the gradient values in a voxel window centered on the edge voxel and oriented in the direction of the surface normal approximation.

A comparison of the two methods is then performed, and the authors find the Deriche algorithm to perform slightly better than Canny on almost every tested sample in terms of measurement precision, but the improvement was particularly visible for flat surfaces. An interesting result is that the authors claim that thanks to their sub-voxel refinement method, "the material transition point can be localized to within 0.01 voxels", but this claim is not reflected even in the best cases in their results section.

Conclusion of the comparison between the methods

- Deriche is slightly better than Canny, but particularly in the case of flat surfaces.
- The sub-voxel refinement method for the Deriche algorithm can refine the surface in diagonal directions while the method based on Canny can only refine in the orthogonal directions. This attenuates the error induced by approximating the surface normals.
- Deriche performs better in multi-material situations where the attenuation coefficients are similar.
- The α parameter in the Deriche operator can be tuned to balance the trade-off between localization accuracy and noise suppression.

1.11 Surface Extraction Using Analytical Gradient Of CT Reconstruction Formula

Presented by Nagai et al. [19] in 2019, the article named "*Accurate surface extraction on CT volume using analytical gradient of FDK formula*" suggests using gradient values to determine the surface's position accurately. Contrarily to the methods presented in section 1.10, this novel approach suggests computing the gradient during the CT reconstruction algorithm, instead of computing the gradient of the 3D image resulting from the CT reconstruction. Doing so offers the following advantage : the CT reconstruction algorithm mentioned in the paper implements the Filtered Back Projection formula, which can be derived analytically to compute the gradient. In contrast, computing the gradient after CT reconstruction involves computing over a voxel grid, introducing discretization errors. Therefore, computing the gradient at the reconstruction step should lead to better results as no discretization artifact is introduced.

The results section of the article shows a roughly 33% accuracy improvement from using this method over the one where the gradient is computed after CT reconstruction. However, we will not delve into the implementation details of this approach as it suffers from a consequent limitation: it requires the CT reconstruction algorithm to be modified, which does not fit the scope of this master's thesis.

1.12 Conclusion

There exist many different methods, each with its strengths and drawbacks. While less accurate but faster methods based on the marching cubes algorithm are preferred in the medical field, gradient-based methods are more commonly used in the industrial field. Since we could not find a consensus on a universally superior method, the next chapters will be dedicated to evaluating the performance of each of the presented methods.

Chapter 2

Experimental Study Of Isosurface Reconstruction Algorithms

In this chapter, we carry out an experimental study of some of the surface extraction algorithms that seemed promising based on our study of the state of the art for surface reconstruction. In particular, we choose to expand the study proposed by R. Greffe by reusing the algorithms that had already been investigated in his own master's thesis [4] and applying those algorithms to input volumes acquired via both circular and helical geometries, to determine whether using a more complex geometry of acquisition can lead to satisfactory results in term of quality of the reconstructed surface. These algorithms were presented from Section 1.2 to Section 1.7.

To this end, the first section of the chapter describes the input data from which a surface will be extracted. The second section describes the software library that implements the surface reconstruction algorithms which are tested in this chapter. The third section describes the evaluation methodology used to assess the reconstruction quality. In the fourth section, we present the results obtained following the test methodology. The fifth and sixth sections describe two techniques designed to improve the quality of the results.

2.1 Description Of The Data

The input data from which a surface will be extracted is a 3D image obtained via CT reconstruction. This CT reconstruction would normally take place after placing the object to analyze inside a CT scanner. In our case, however, the CT scanning is simulated using a software library using a modeled object as input. This way, we can simulate different geometries of acquisitions without the need to have a CT scanner able to perform those acquisition geometries. Moreover, some artifacts inherent to the properties of a physical scanner will not be present in a simulation, allowing us to focus on the artifacts that are relevant to the problem we are trying to solve.

It is important to choose an object to scan that presents interesting properties. Indeed, we would like to assess the performance of the surface reconstruction algorithms both on flat and curved surfaces, but also around edges. To this end, we choose to work with the modeled object illustrated on Figure 2.1, as it presents the following geometrical properties :

- Flat surfaces on the sides of the cubic element, which are parallel to the planes containing the cartesian {x,y,z} axes.
- Flat surfaces on the sides of the pyramidal element, which are not parallel to the planes containing the cartesian {x,y,z} axes.
- A cylindrical hole inside the cubic element, to evaluate the performance on a circular curvature, as it is often found on industrial items.

- Sharp edges between the different faces of the object.

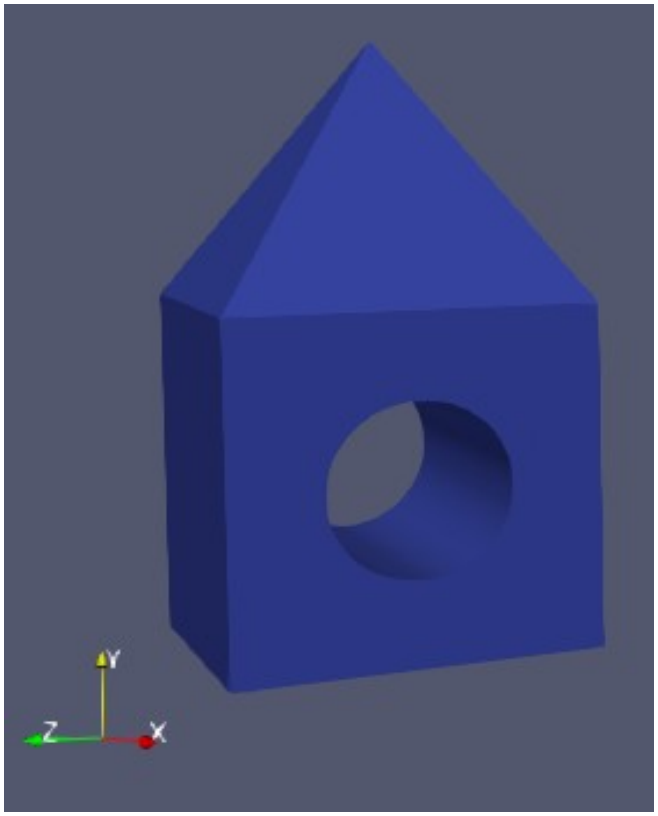


Figure 2.1: Modeled object consisting of a pyramid mounted on a cube that features a cylindrical hole.

The dimensions of the object are as follows :

- Each side of the cubic element is 40mm wide.
- The height of the pyramidal element measures 30mm.
- The radius of the cylindrical hole inside the cube is equal to 10mm.

A bounding box surrounding the object would therefore be dimensioned as 40mm x 40mm x 70mm.
For this experiment, the resolution used to simulate the CT scanning is 200µm/voxel (or 0.2mm/voxel). Therefore, a bounding box that does not take into account the air around the object would be dimensioned as 200 voxels x 200 voxels x 350 voxels.

Both a circular and a helical acquisition geometries are then chosen for the CT scan simulation. The 3D images, or volumes, resulting from the CT reconstruction are coded in 16 bits, meaning that the pixel values of these images, which are the input of the surface extraction algorithms, range from 0 to 65,535. These voxel values represent the local density of the material.

2.2 Description Of The Software Libraries

To avoid re-implementing all the surface extraction algorithms, we decided to use the open-source software library named *VTK*, short for *Visualization Toolkit*. Our choice is motivated by the following reasons :

- It provides an implementation for all of the isosurface extraction algorithms that we plan to analyze in this chapter.

- It allows for high-performance computing as most of its functionalities leverage the capabilities of parallelism whenever possible.
- It is structured as an easy-to-use pipeline where each class is represented as a "filter", whose output can very easily be used as input for another filter. This is very practical as multiple image transformations can be chained one after the other. Moreover, the library implements a lot of filters that will prove useful for our experiments.

VTK is designed to work with 3D representations. As such, it uses the *cell* as its most basic form of data representation. A *cell*, which can be triangular, square or polygonal, represents a set of points in space linked together by edges. A collection of these cells is called a *polydata*, and is the output data representation of the surface extraction algorithms that we will be testing.

As the input data is a 3D image instead of a *polydata*, another open-source library is required to bridge the input with an "equivalent" *polydata* representation. The library we chose for this is called *ITK*, short for *Insight Toolkit*, as it features a method to smoothly perform the transition from image to *polydata*. Moreover, it possesses many image filters that will be useful for our experiments. For example, it allows us to perform some pre-processing on the input image to remove unnecessary space around the object, leading to smaller image size and shorter processing time.

2.3 Evaluation Methodology

To evaluate the quality of reconstruction of the output surfaces, we compare them to a reference surface model. In our case, this reference surface is stored as a *stl* file and is visually identical to the model illustrated on Figure 2.1. To measure the deviation between an extracted surface and the reference surface, we compute the distance between each point of each *cell* composing the *polydata* representing the extracted surface and its closest point on the reference surface.

We then use a property of the *polydata* data structure which consists in adding scalar values to a set of points or *cells*. This way, the distance between an extracted point and its projection on the reference surface can be added as a scalar value to the extracted point. This strategy allows us to obtain a visual representation of the distribution of the error over the entire set of points constituting the extracted surface. While this does not yet yield aggregated quantitative results, having access to some visual representation of the error is useful to perform a qualitative analysis of the extracted surface. As such, several instances of results using this visual representation will be used throughout the next sections.

To perform a quantitative analysis of the results, we plot the normalized distribution of the error measured in voxel size. We chose to display the results in voxel size because one of the objectives of this experiment is to determine whether these algorithms that were simply taken from an open source library would be sufficiently accurate to achieve metrology precision, which is decided to be equal to 10% of the length of a voxel.

It is worth noting that the error metric, measured as the distance between an extracted point and its projection on the reference surface, is unsigned. While using signed error metrics is useful to determine the locations where the points have a tendency to be reconstructed inside or outside the reference surface, unsigned error measurements allow for easier-to-read distribution graphs and are more relevant when assessing whether the results satisfy the metrology criterion.

We also add some descriptive statistics on top of the distribution, such as the maximal error value, the mean error, the standard deviation of the error and the percentage of points that satisfy the metrology criterion.

2.4 Surface Extraction Results

The distributions and their corresponding descriptive statistics are presented in Figure 2.2.

A first general observation is that the distribution of the error always presents two distinct regions of high frequencies, independently of the reconstruction geometry. A first peak of frequencies is located around the vertical red line that symbolizes an error of the tenth of the size of a voxel, corresponding to the metrology criterion. The points corresponding to these levels of errors are mainly located on the planar faces of the object, which are, as expected, easier to reconstruct. The second region of high frequencies occurs on the far right of each graph, where a distinct peak of frequencies can be observed. It mainly corresponds to the points belonging to the bottom face of the object. One explanation for this phenomenon could be that the input image was of lower quality at this location following the CT reconstruction process.

Another general observation is that the proportion of points whose associated error satisfies the metrology criterion is always higher for the circular geometry than for helical geometry. All other statistical measurements, such as the maximal error, mean error, and standard deviation of the error, are consistently better in helical geometry than in circular geometry. This intriguing result might be explained by the fact that using helical acquisition geometry improves the overall quality of the input image, but that specific locations such as the bottom face of the object are of even worse quality, which could also explain why the peak of frequencies on the far right of each graph is much more pronounced in helical geometry.

No specific algorithm stands out in terms of the quality of the reconstructed surface, as the error statistics are relatively similar for each of the algorithms in helical geometry.

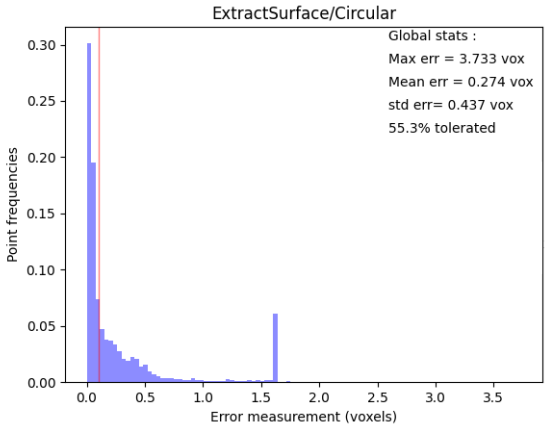
In terms of the execution speed, the performance of each surface reconstruction algorithm is displayed in Table 2.1. The *Surface reconstruction from unorganized points* algorithm stands out as it takes hundreds of seconds to complete, even with a neighborhood count as low as 12. This is unsatisfactory, as we require the surface reconstruction process to complete in a few tens of seconds at most. This behavior was somewhat expected, since we had listed the long execution time among the disadvantages of this method in our state of the art, but we were hopeful that using a modern computer would have improved this aspect, since the paper describing the method was published in 1992.

The *SurfaceNets* and *FlyingEdges* algorithms stand out as well due to their very low execution times compared to the other algorithms. This behavior was expected since these two methods make use of parallelization to speed up the process, but it is worth noting that the quality of the reconstructed surface does not appear to be negatively impacted, except that staircase-like structures can be seen on the planes that make up the pyramidal faces of the object, as shown on Figure 2.4.

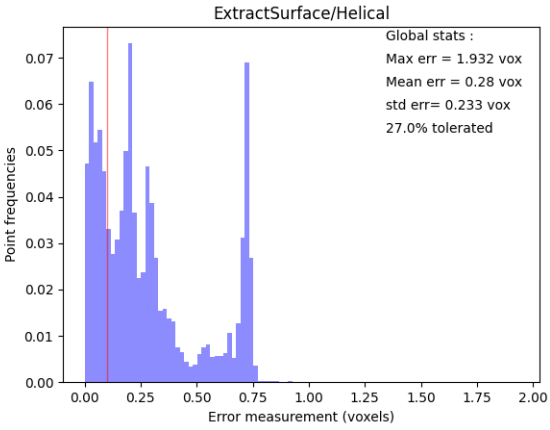
Another remarkable result is that edges seem to be better preserved using the *PowerCrust* algorithm, which makes sense since this method implements a specific feature that allows for sharp edges reconstruction, as shown in the state of the art on Figure 1.5. However, it was quite unreliable in the sense that its parameters had to be tuned very carefully for it to produce any output, which is unsatisfactory considering that we expect our surface reconstruction technique to have low sensitivity to the shape of the object. Figure 2.3 emphasizes how the *PowerCrust* algorithm reconstructs sharper edges than the *Poisson* algorithm, and how the surface extracted by both algorithms still contains artifacts.

2.5 Iterative Closest Point Algorithm

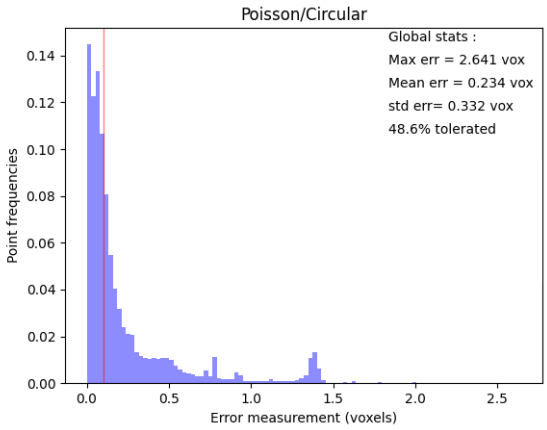
The *polydatas* analyzed in the previous sections consist of *cells*, which are collections of points in space. When looking at the bounding box of the set of all such points, we notice, as expected, that it does not exactly coincide with the bounding box of our reference surface. We want to make sure that our reconstructed surface and reference surface are oriented and positioned similarly in space to ensure a fair



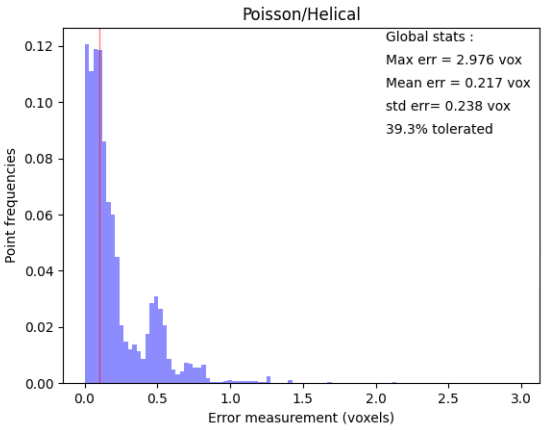
(a) Extract Surface with circular geometry



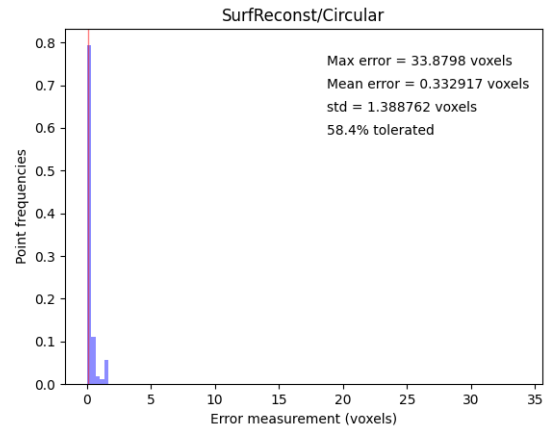
(b) Extract Surface with helical geometry



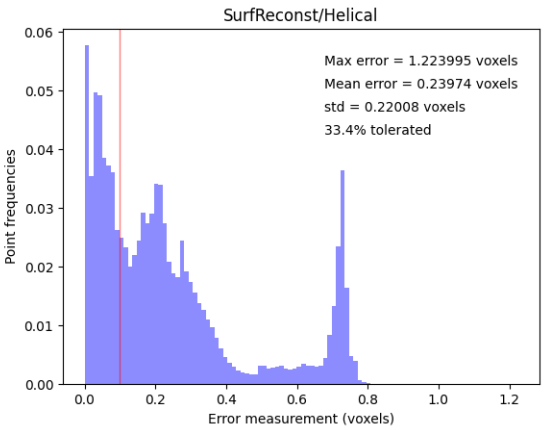
(c) Poisson with circular geometry



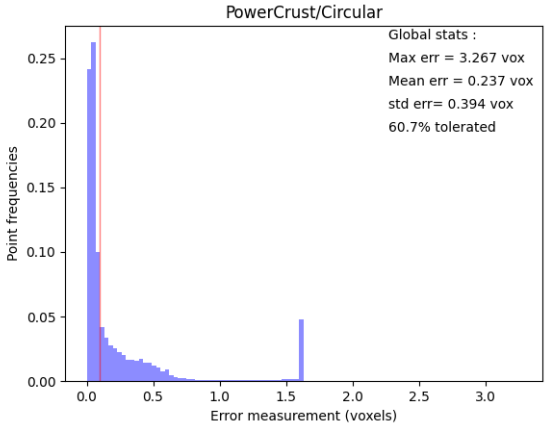
(d) Poisson with helical geometry



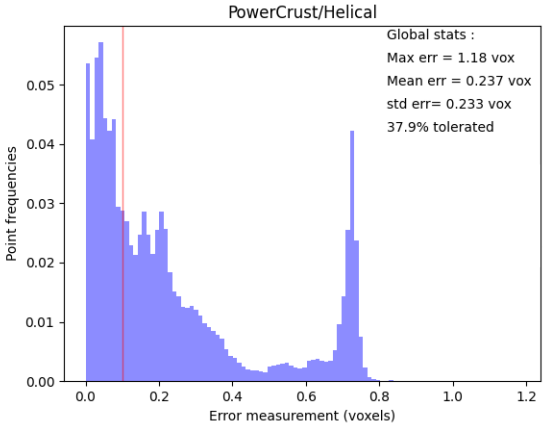
(e) SurfaceReconstruction with circular geometry



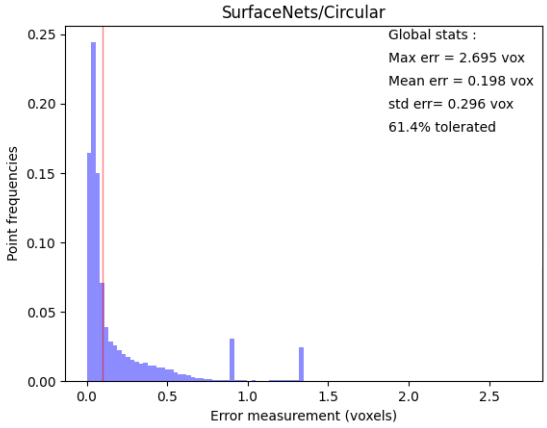
(f) SurfaceReconstruction with helical geometry



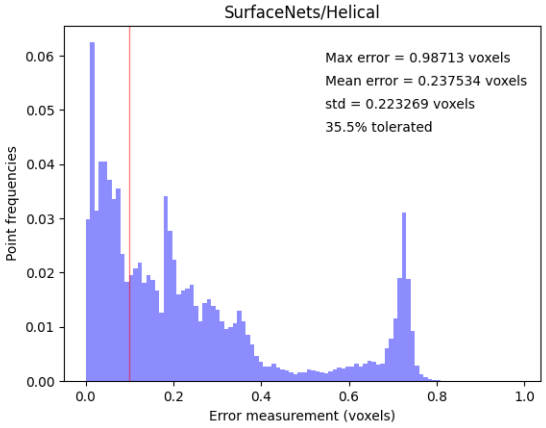
(g) PowerCrust with circular geometry



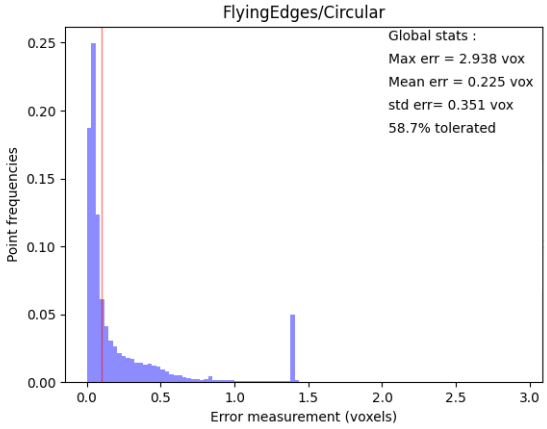
(h) PowerCrust with helical geometry



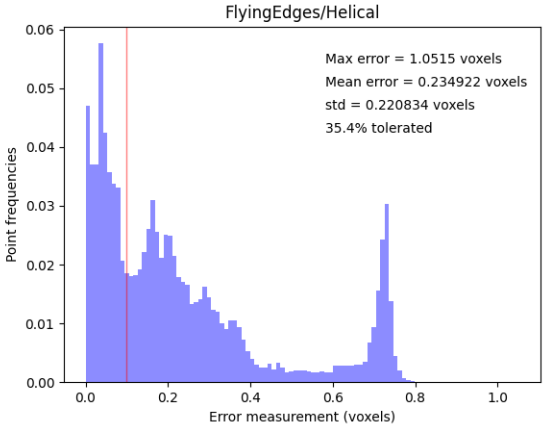
(i) SurfaceNets with circular geometry



(j) SurfaceNets with helical geometry

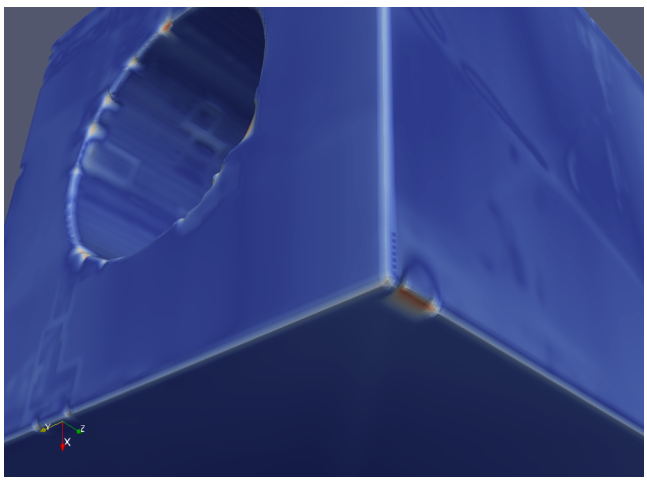


(k) FlyingEdges with circular geometry

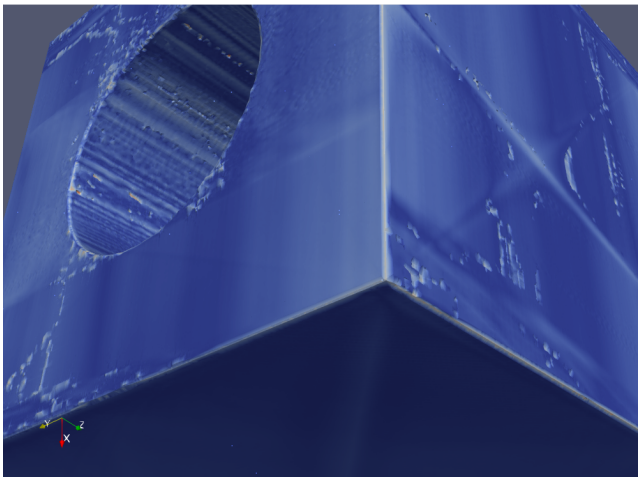


(l) FlyingEdges with helical geometry

Figure 2.2: Histograms of the unsigned measured error in voxels at every point between the surface generated by the different algorithms and the reference surface. For each algorithm, the results for both circular and helical geometries are displayed. The vertical red line that is located at x = 0.1 represents the maximal desired error to be under the metrology criterion. Descriptive statistics for the entire set of points are also displayed.



(a) Surface extracted using the Poisson algorithm.



(b) Surface extracted using the PowerCrust algorithm.

Figure 2.3: Surface extracted using two different algorithms. On the left, the edges are not well defined, especially around the cylinder opening, but the planar faces are smooth. On the left, the edges are sharply defined, but there are lots of artifacts on the planar faces.

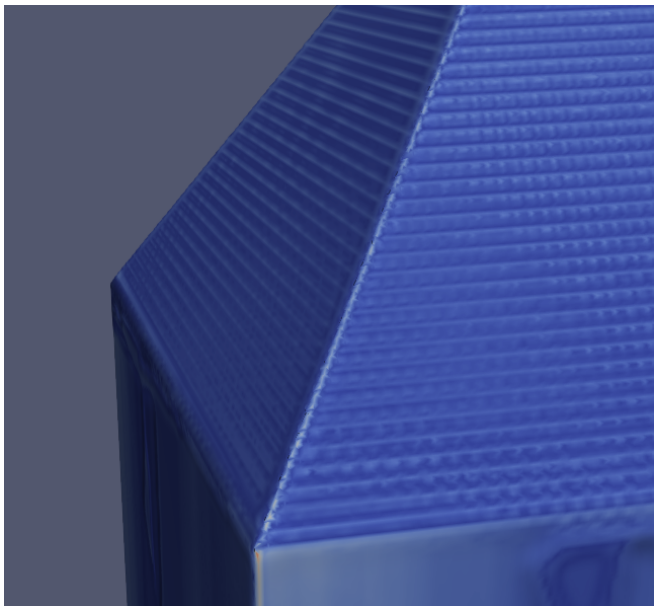


Figure 2.4: Surface extracted using the SurfaceNets algorithm.

Algorithm	Time (ms)
ExtractSurface	8,099
Poisson	7,256
PowerCrust	10,654
SurfaceReconstruction	767,165
SurfaceNets	65
FlyingEdges	1,680

Table 2.1: Execution times of the surface reconstruction algorithms in milliseconds.

comparison. To this end, we investigate an algorithm called *Iterative Closest Point*, which is aimed at minimizing the distance between two sets of points, or between a set of points and a set of planes. To perform this minimization, several types of transformations can be applied to the dataset, such as a rigid body transform or an affine transform. The rigid body transform is relevant to our experiment as it does not modify the relationship between the points since it only consists of a rotation followed by a translation of the points.

First introduced by Chen et al. [20], the purpose of the algorithm is to find the rotation/translation pair that minimizes the mean square point-to-point distance metric. As it is iterative, it is designed to work according to the following steps :

1. For each point of the set of points to align, match the closest point in the reference point cloud or the closest point on the planes in the reference set of planes. This pair of points is then stored.

2. An error function is computed as the sum of the Euclidean distances of all the matching point pairs. Then, a singular decomposition value is performed to minimize this error function and obtain the rotation and translation matrices corresponding to the optimal alignment.

3. Apply the rotation and translation matrices obtained at the previous step to the set of points, and reiterate steps 1 to 3 to achieve convergence.

One significant drawback to this algorithm is that it may converge to a local minimum that does not correspond to the transformation that would lead to a global optimization. For instance, our experiments showed that the extracted surface and the reference surface needed to be roughly aligned manually before applying the ICP algorithm to avoid this phenomenon from occurring.

In terms of complexity, here is how the different steps scale :

- To find every closest-points pair (n, m) with $n \in N$ and $m \in M$, where N is the number of points in the set of points to align and M is the reference set of points, the complexity is $O(NlogM)$ thanks to the use of a k-d tree that allows for spatial hashing. The construction of this k-d tree is $O(M)$.

- The singular value decomposition step involved in finding the optimal rotation and translation matrices is computed in $O(N)$, where N is the number of points in the set of points to align.

In practice, 10 to 50 iterations are required to obtain enough convergence. If we denote K the number of iterations, the total complexity of the algorithm is $O(K(NlogM + M))$.

2.6 Bilateral Filter

Some of these algorithms, such as the PowerCrust, require a point cloud as input, which we decide to extract from the image using a Canny edge detection filter. Other algorithms, such as the SurfaceNets, take label maps as input, which we also decide to extract from the input image using a Canny edge detection filter. The first step of the Canny filter consists of applying a Gaussian filter to the image to

attenuate its noise. However, our reconstructed object displays sharp edges, like most industrial parts, which may lose sharpness after the Gaussian filter is applied. Therefore, we choose to replace the Gaussian filter in the Canny pipeline by a *bilateral filter* [21], which is an edge-preserving smoothing filter. The bilateral filter can be formulated as

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(||p - q||) G_{\sigma_r}(|I_p - I_q|) I_q,$$

where σ_s controls the spatial extent of the kernel used for spatial smoothing, identically to a Gaussian filter, and where σ_r controls the range kernel used for smoothing differences in pixel intensities. The more σ_r increases, the more the filter approaches Gaussian convolution. Around an edge, neighboring pixels display large intensity differences, leading to lower weights given by the Gaussian function G.

2.7 Results After Improvements

In this section, we perform the experiment we carried out in Section 2.4 a second time, this time applying the ICP and bilateral filter improvement techniques. Figure 2.5 shows the distributions and their corresponding descriptive statistics, after a rigid body transform is applied to the reconstructed surface to align it with the reference, and with bilateral filtering when relevant.

A first observation is that applying the ICP algorithm helped remove the peak of high frequencies that we associated with the bottom face of the object. It would make sense, since the translation applied to the reconstructed surface will lift it to match better with the reference surface. This has the effect of significantly increasing the proportion of points satisfying the metrology criteria for helical geometry, as some of the algorithms now reach 50% to 60% of tolerated points. These algorithms, namely the *Poisson*, *ExtractSurface*, *SurfaceReconstruction*, and *PowerCrust* algorithms, also display improved descriptive statistics, with a mean error almost under the metrology criterion and a lower standard deviation.

We also observe that the *SurfaceNets* and *FlyingEdges* algorithms perform much worse than before the application of ICP, most likely due to a failure of ICP to find a global minimum.

2.8 Conclusion

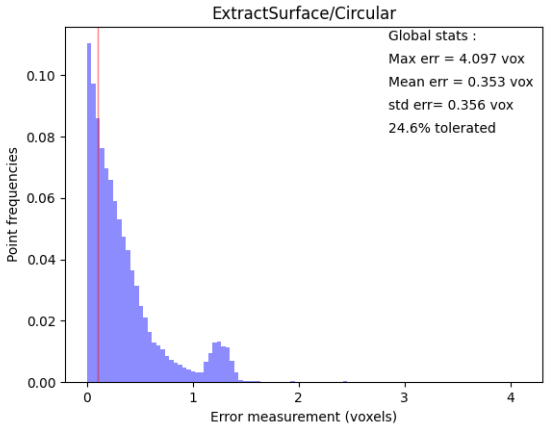
Overall, there was no clear difference in the quality of the reconstructed surface between the methods based on the marching cubes algorithm and those that were not.

Two methods were considered unsatisfactory as they required too much parameter tuning or took too long to complete.

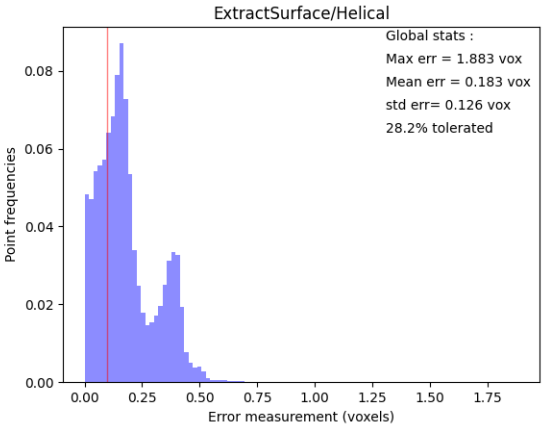
Using input data acquired via helical geometry seemed to improve the overall accuracy of the reconstructed surface while also worsening the proportion of points that satisfy the metrology criterion.

We came up with two techniques aimed at improving the quality of the output surface: ICP and bilateral filter. While these techniques significantly improved the performance of some algorithms, they decreased the performance for others, leading us to believe that these techniques can have a positive impact but are unreliable.

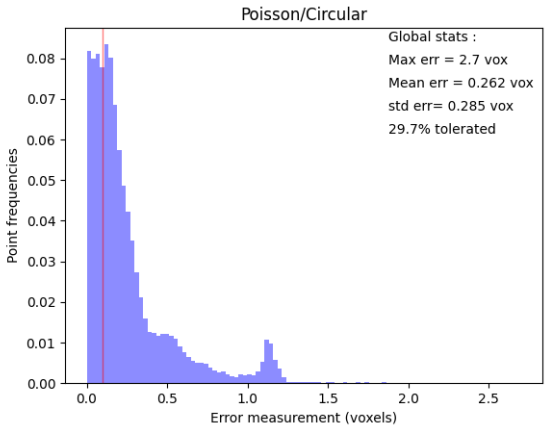
In conclusion, using data acquired via helical geometry may lead to more accurate results, which was expected since it is supposed to be burdened by fewer CT reconstruction artifacts. However, none of the algorithms that we tested demonstrated results that satisfied the metrology criterion.



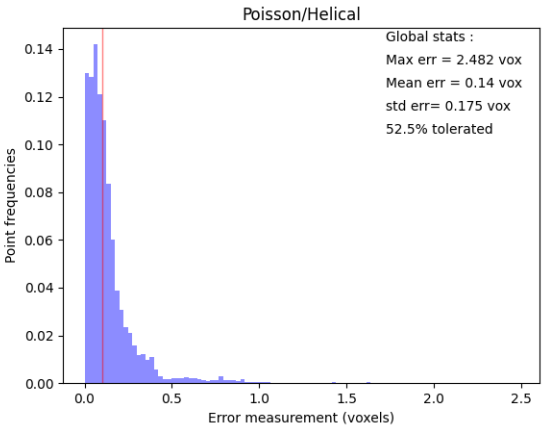
(a) Extract Surface with circular geometry



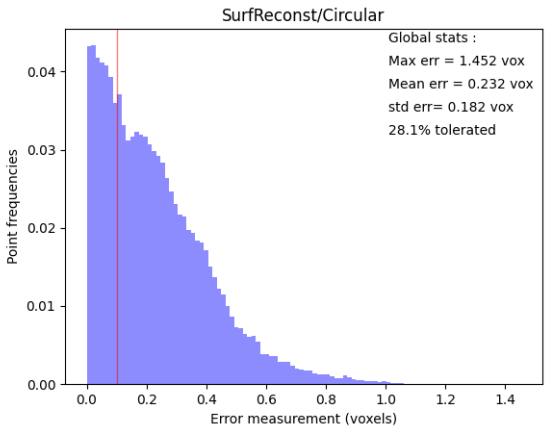
(b) Extract Surface with helical geometry



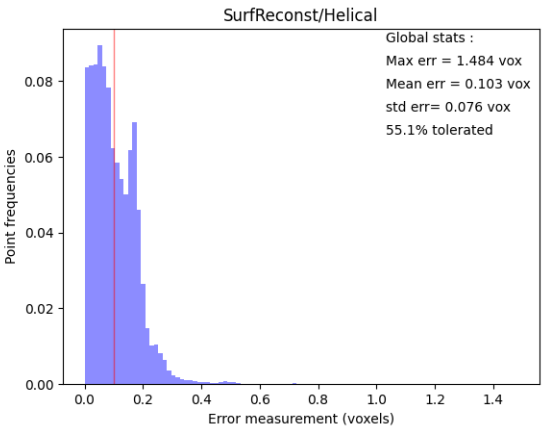
(c) Poisson with circular geometry



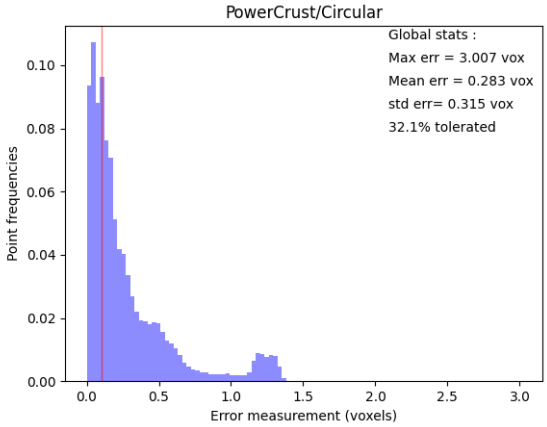
(d) Poisson with helical geometry



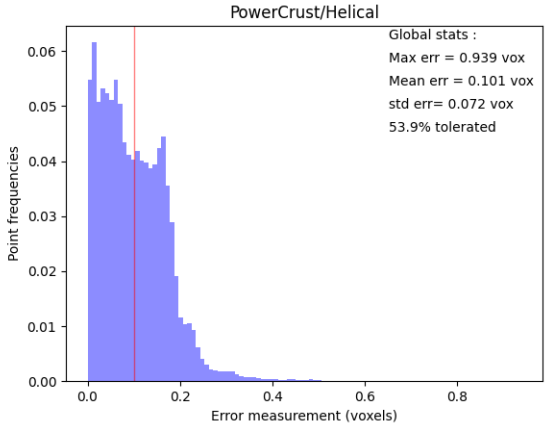
(e) SurfaceReconstruction with circular geometry



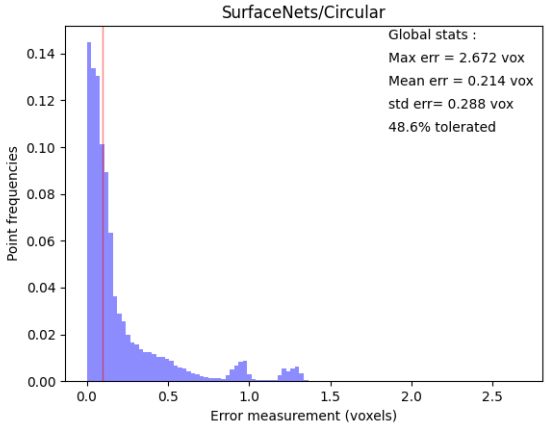
(f) SurfaceReconstruction with helical geometry



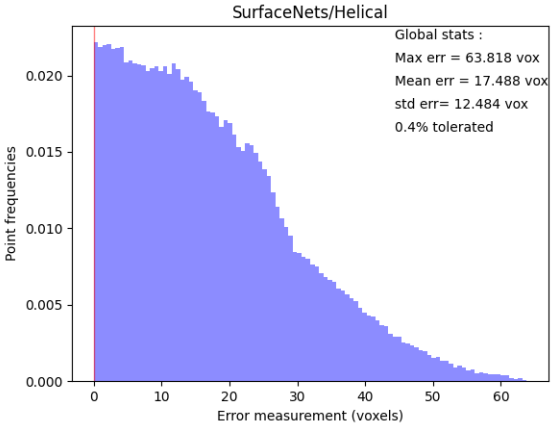
(g) PowerCrust with circular geometry



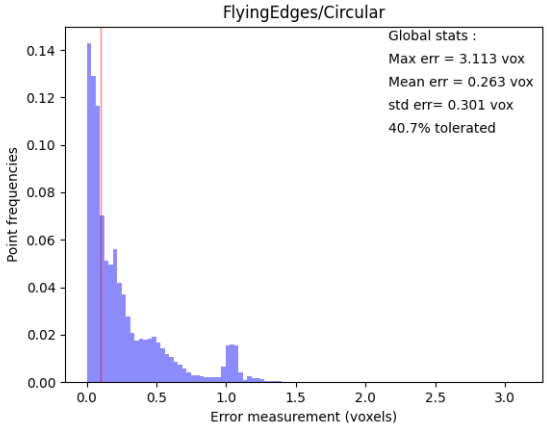
(h) PowerCrust with helical geometry



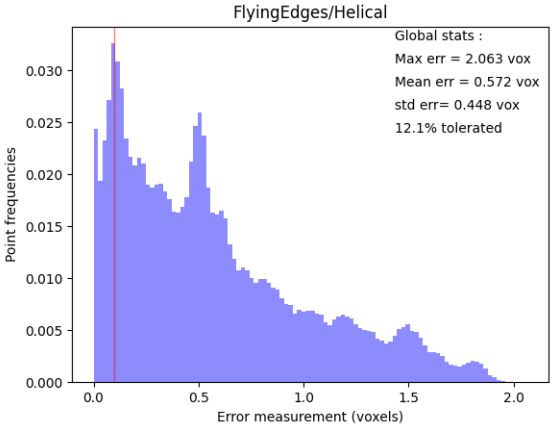
(i) SurfaceNets with circular geometry



(j) SurfaceNets with helical geometry



(k) FlyingEdges with circular geometry



(l) FlyingEdges with helical geometry

Figure 2.5: Histograms of the measured unsigned error in voxels at every point between the surface generated by the different algorithms and the reference surface, after alignment using the Iterative Closest Point algorithm and bilateral filtering wherever relevant. For each algorithm, the results for both circular and helical geometries are displayed. The vertical red line that is located at x = 0.1 represents the maximal desired error to be under the metrology criterion.

Chapter 3

Sub-Voxelic Refinement Algorithms

In chapter 2, we experimented with several surface extraction algorithms that were already implemented in the VTK software library, which also expanded the study of R. Greffe [4] to helical geometries.

In this chapter, the goal is to investigate methods based on discontinuity. In particular, two methods that we discussed in Section 1.9 and Section 1.10 in the state of the art showed promise for industrial applications and metrology in particular. In the following sections, we implement those methods and evaluate their quality with regard to the metrology criterion.

A description of the pipeline used throughout this experimentation is first presented in Section 3.1. Section 3.2 and Section 3.3 are dedicated to the implementations of the methods based on discontinuity, as well as a discussion of the results. Finally, Section 3.4 presents an analysis of the design choices and parameter choices that influenced the obtained results.

3.1 Description Of The Pipeline

Figure 3.1 shows a schematic representation of the pipeline followed throughout this experiment. The input of the pipeline is a 3D image obtained via CT reconstruction, acquired via helical geometry only as we are not looking to compare circular and helical geometries like we did in chapter 2, but instead trying to evaluate the potential accuracy of this method in the optimal conditions. This image is then passed as input to a first surface approximation filter, such as Canny [17], which defines the surface as a set of voxels based on local maxima of the gradient values of the voxels. A point cloud is extracted from this set of voxels, where each point is chosen as the center coordinates of each voxel. This point cloud now undergoes sub-voxelic refinement, in order to precise the location of the points so that they do not correspond to voxel centers anymore. This step, highlighted in red on Figure 3.1, is the main subject of this chapter's experimentation. As such, the methods described in the following sections refer to this step in the pipeline. After the sub-voxelic refinement step, the refined point cloud is passed through a surfacing algorithm, such as the PowerCrust method that we described in section 1.7 and experimented on in chapter 2. From this surfacing algorithm, the final surface is generated as output in *polydata* or *stl* format.

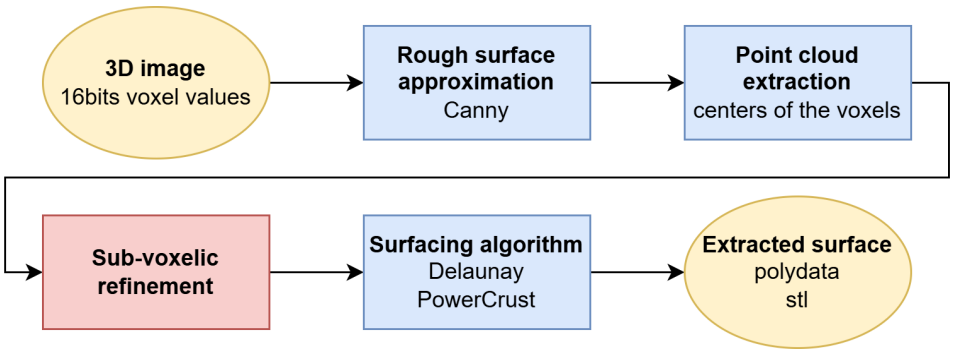


Figure 3.1: Illustration of the pipeline used for surface extraction for methods that take a point cloud as input. Text in bold represents the conceptual processes, while regular text describes some examples of implementation of these concepts.

3.2 Center Of Mass

The idea behind this center of mass refinement method, described in detail in Section 1.9, is to approximate the location of the object's surface by the location of the discontinuities in the image's voxel values.

As such, for each point in the point cloud obtained via the Canny edge detection technique during the preliminary surface extraction step, the method consists in averaging the density values of the voxels located in a neighboring window centered on the point position. During the averaging procedure, the density values are weighted by their distance to the point to simulate a "center of mass" or "center of density". This center of mass is computed in all three Cartesian directions, and the point is refined by modifying its x, y, and z coordinates by the corresponding centers of mass.

Algorithm 1 shows a high-level description of how the method has been implemented.

Algorithm 1 Point Refinement Via Center of Mass of Density Values

```
for each point p in point_cloud do
    for each direction d in {x, y, z} do
        Initialize sum_pos[d] ← 0, sum_weights[d] ← 0
        for each voxel in NeighborhoodWindow(p) do
            g ← voxel.density
            sum_weighted_pos[d] ← sum_weighted_pos[d] + g · voxel.position[d]
            sum_weights[d] ← sum_weights[d] + g
        end for
        center_of_mass[d] ← sum_weighted_pos[d]/sum_weights[d]
    end for
    RefinePointPosition(center_of_mass, p)
end for
```

Results

The resulting distribution of the unsigned error after using Algorithm 1 as sub-voxelic refinement method is presented in Figure 3.2. The procedure took 863 ms to refine 296,631 points.

These results are not very promising as they do not show any improvement compared to the results obtained in the previous chapter. This was to be expected, as the method is very simple and only takes into account the gray values of the pixels.

The execution time was pretty low, however.

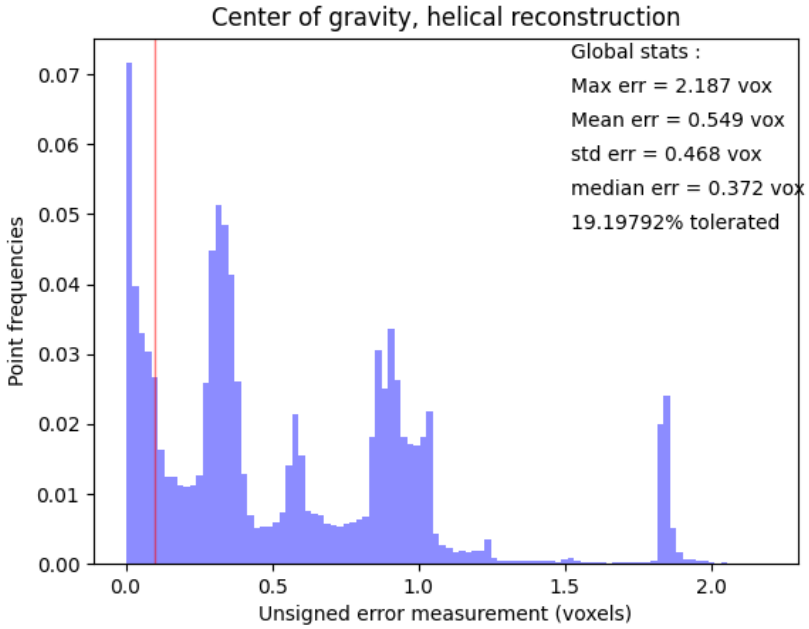


Figure 3.2: Histogram of the unsigned error measured in voxel size for every point of the point cloud obtained after sub-voxel refinement using the center of mass method. The vertical red line located at x = 0.1 represents the maximal error under which the metrology criterion is satisfied.

3.3 Gradient-Based Algorithm

This method, described in more detail in Section 1.10, consists in using the positions of the local maxima of the gradient of the image's voxel values to approximate the locations of the surface.

Given a point cloud as input, this method can be described by the following sequence of steps :

1. Compute the gradient magnitude for each voxel in the input image to obtain a gradient image.
2. For each point in the point cloud, determine which of the main directions maximizes the gradient magnitude. By main directions, we mean the set of directions which consists of: the three Cartesian directions x, y, and z, the 6 directions that bisect two Cartesian axes, and the 4 directions that pass by the corners of the voxel. These main directions are approximations of the surface's normal, and as such, the point will be refined in the main direction that maximizes the gradient's value. Figure 3.3 shows a voxel with its 13 associated main directions.
3. Refine the position of the point in the main direction chosen at the previous step. The following methods are proposed :
   - Compute the center of mass of the gradient values in a neighborhood voxel window centered on the point and offset the coordinates accordingly in the chosen main direction.
   - Construct a cubic spline model to accurately interpolate the gradient values of the voxels in a neighborhood window centered on the point and oriented in the chosen main direction. Use this model to find the location that maximizes the gradient's value, and refine the point's coordinates accordingly in the chosen main direction.

A pseudocode summarizing the steps described previously is provided below. In particular, Algorithm 2 describes the variant where a center of mass of the gradient's value is used to refine the point's position, while Algorithm 3 describes the variant where a cubic splines model is used to interpolate and find the maximum of the gradient's value.

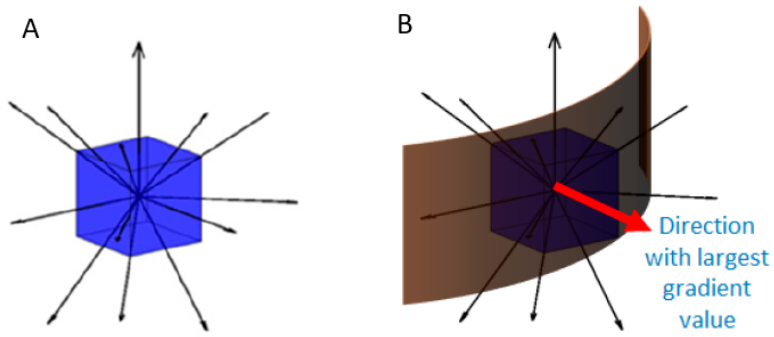


Figure 3.3: A) An illustration of the 13 main directions of a voxel. B) The direction of the largest gradient value is an approximation of the surface's normal. Reproduced from Ontiveros et al., "Analysis of Surface Extraction Methods Based on Gradient Operators for Computed Tomography in Metrology Applications" [18], 2018

Algorithm 2 Gradient-Based Point Refinement With Center Of Mass

```
gradient_image ← Compute_Gradient(volume)
for each point p in point_cloud do
    max_dir ← FindMainDirectionThatMaximizesGradientValue(gradient_image, p)
    Initialize sum_weighted_pos ← 0⃗, sum_weights ← 0
    for each voxel in NeighborhoodWindow(max_direction, p) do
        g ← voxel.gradient_magnitude
        sum_weighted_pos ← sum_weighted_pos + g · voxel.position
        sum_weights ← sum_weights + g
    end for
    center_of_mass ← sum_weighted_pos/sum_weights
    RefinePointPosition(p, center_of_mass, max_dir)
end for
```

Algorithm 3 Gradient-Based Point Refinement With Cubic Splines Interpolation

```
gradient_magnitude_image ← ComputeGradientMagnitude(input_volume)
for each point p in point_cloud do
    max_direction ← FindMainDirectionThatMaximizesGradient(gradient_magnitude_image, p)
    Initialize cubic_spline
    for each voxel in InterpolationWindow(max_direction, p) do
        cubic_spline.AddGradientValue(voxel.position, voxel.gradient_magnitude)
    end for
    offset ← FindMaxGradientValue(cubic_spline)
    RefinePointPosition(p, offset, max_dir)
end for
```

Results

Figure 3.4 shows the distribution of the unsigned error after using Algorithm 3 as sub-voxelic refinement method, using the same input data that was described in section 2.1 and the evaluation methodology described in Section 2.3. The process took 2033 ms to refine 296,631 points.

An immediate observation is that these results do not show significant improvements in terms of accuracy compared to the methods presented in Chapter 2. Approximately only 30% of the refined points fall under the metrology criterion, which is worse than the best methods evaluated previously. The maximal error is over the size of a voxel, and the mean error is comparable to the mean error of the best methods evaluated previously.

In conclusion, these results do not allow us to conclude that the gradient-based method in its current form is better in terms of accuracy than the algorithms evaluated in Chapter 2. Moreover, we reviewed in Section 1.10 the article that presented this method, in which the authors claim that "the material transition point can be localized to within 0.01 voxels", even though their results do not seem to back that claim. Similarly, our results tend to disprove this claim.

It is also worth noting that these results are more promising than the results we had obtained in Section 3.2 for the center of mass method. It confirms our intuition that a method based on the gradient of the image gray levels would perform better than a method working with the gray levels directly.

Nevertheless, this method has the advantage of being more flexible than the methods evaluated in chapter 2 since we can control its implementation. As such, the next section is dedicated to an experimental study of the different sub-methods and parameters that are involved in this sub-voxelic refinement method.

3.4 Parameter Study

During our research on the state of the art for surface reconstruction algorithms, we discovered that the gradient-based method is among the most commonly used for metrology applications in the industrial domain, as its supposedly allows for high accuracy in the localization of the surface. Despite these promising claims, the results we obtained in the previous section are still far from satisfactory in terms of surface accuracy.

In an effort to optimize these results and to get a better understanding of the phenomena at play, we perform an analysis of the different parameters and mathematical models involved in the gradient-based refinement method. In particular, we distinguish three potential sources of inaccuracy :

- The choice of operator used to compute numerically the gradients of the image's intensities.
- The method used to determine the location of the surface once the gradient has been computed.
- The choice of constraining the surface normal's orientation to one of the 13 main directions as illustrated on Figure 3.3.

To accurately evaluate the impact of each of these potential sources of inaccuracy, it is necessary to work with perfect data, as the noise and artifacts that necessarily come with CT images are sources of inaccuracy. To this end, and for the sake of simplicity, we decided to generate perfect voxelizations of simple 2D geometric shapes. In particular, we voxelized a 2D square and a 2D disk by computing the exact analytical portion of each voxel that is enclosed within the square or the disk, effectively representing the density of the material inside those voxels. These shapes can also be translated relative to the origin of the reference frame in order to obtain different density values in each voxel. An example of this voxelization is shown on Figure 3.5.

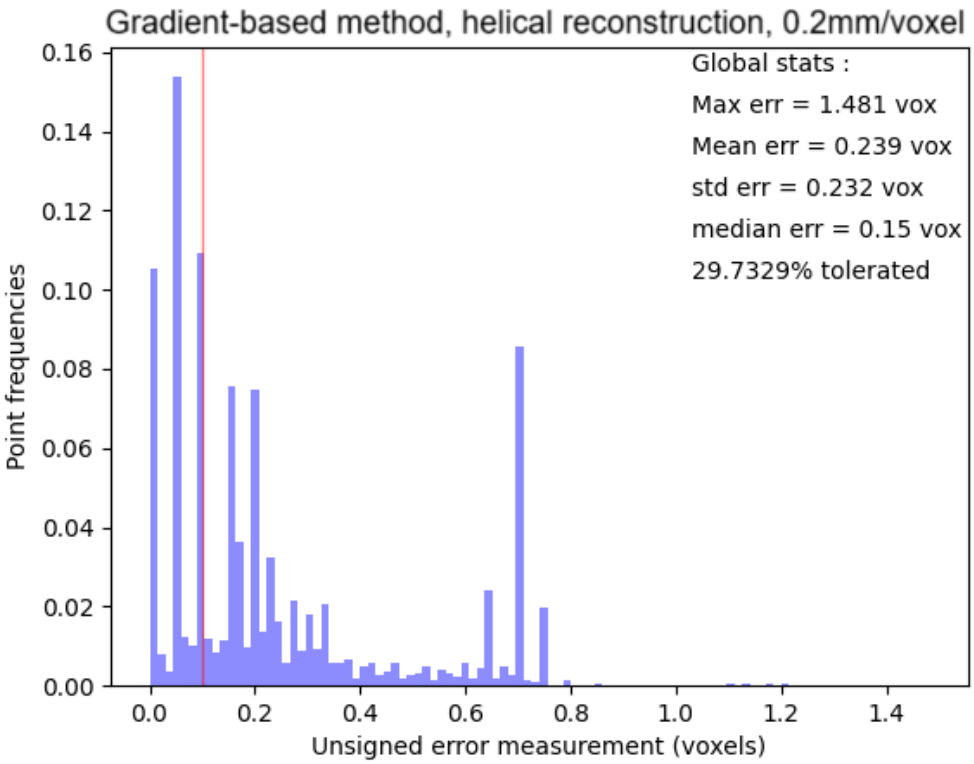


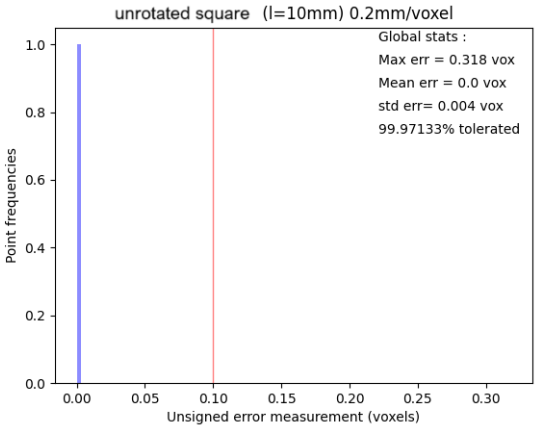
Figure 3.4: Histogram of the unsigned error measured in voxel size for every point of the point cloud obtained after sub-voxel refinement using the gradient-based method. The vertical red line located at x = 0.1 represents the maximal error under which the metrology criterion is satisfied.

(a) Voxelized square offset by 0.75 voxel in the horizontal and vertical directions. Side length is 100 voxels.

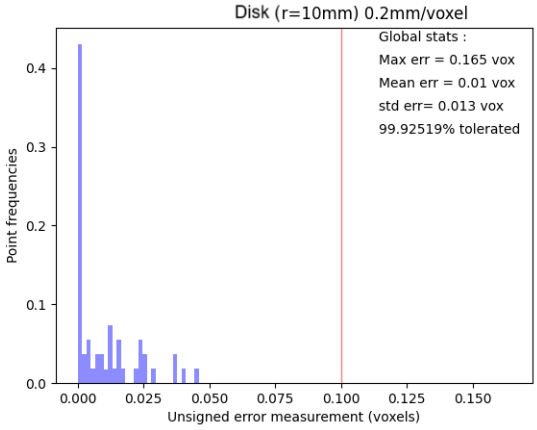
(b) Untranslated voxelized disk. Diameter length is 100 voxels.

Figure 3.5: Examples of non-binary voxelizations of simple geometric shapes.

A natural preliminary step is to apply a 2D implementation of Algorithm 3 to these simple 2D voxelizations to see whether we can obtain a perfect surface reconstruction. In other words, let us first determine whether the poor results obtained earlier were entirely due to the noisy nature of the data or if we were correct in assuming the existence of other error-inducing phenomena.



(a) Voxelized square offset by 0.75 voxel in the horizontal and vertical directions.



(b) Untranslated voxelized disk.

Figure 3.6: Histogram of the unsigned error measured in voxel size for every point of the point cloud obtained after sub-voxelic refinement using the gradient-based method on voxelized 2D simple geometric shapes. The vertical red line located at x = 0.1 represents the maximal error under which the metrology criterion is satisfied.

Figure 3.6 shows that working with noiseless data to perform sub-voxelic refinement leads to much more accurate results, but that there are still some errors that cannot be explained by the quality of the input data. As such, we proceed in the following sections with some investigation of the potential sources of inaccuracy in the design choices of the method.

3.4.1 Gradient Operators

A first design choice in the implementation of the gradient-based refinement method is the operator used to compute the gradient. To this end, we investigate three distinct gradient operators :

- The **Deriche** operator, which is the operator originally chosen in the article describing the gradient-based method [18], described by Equation 1.10. We choose to set the α parameter to 5, as suggested by the authors.
- The **finite central difference**, computed as

$$\delta_h[f](x) = \frac{f(x+h) - f(x-h)}{2h}$$

- The **Sobel** operator [22], commonly used for edge detection, is described as

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

We now apply these three operators to our noiseless 2D data, and use the resulting gradient to localize the surface's position using both the center of mass method described in Algorithm 2 and the cubic

splines interpolation method described in Algorithm 3. The size of the interpolation window is set to 7 voxels, as it is a value that worked well in practice, but that we will experiment with deeper in the next section.

Figure 3.7 shows graphs of the different cubic splines obtained for each gradient operator, where the spline is computed using the gradient values inside an interpolation window centered around the point to refine. In this case, we investigate a point located on a vertical side of a 2D voxelized square that is offset by 0.375 voxels in the x and y directions, and one point located exactly in the corner of the same 2D voxelized square. The dotted vertical lines designate the x position where the interpolated gradient reaches its maximum value, corresponding to the surface's location. The red vertical lines represent the locations of the centers of the voxels surrounding the surface point that is being refined, and the values of the surface locations obtained using the center of mass method are also displayed on the right.

In the case of a point located on the side of the 2D square, there is no difference between the different gradient operators, which yield the same surface localization, as shown on Figure 3.7a. While the cubic squares interpolation method is off by 15%, the center of mass method yields the expected result for the localization of the square's surface.

Figure 3.7b shows that for a point located exactly on the corner of the translated 2D cubes, the central difference operator produces much more accurate results than the other two operators, both for the cubic splines interpolation and center of gravity methods.

The same experiment is then performed on points belonging to the surface of a 2D voxelized disk. Specifically, we investigate the resulting surface localization for each operator and each method for both a point located at a 0° angle on the disk and a point located at a 45° angle in the disk.

For the point located at 0°, the results are shown on Figure 3.8a. Similarly to the case of the 2D square, it seems like the different gradient operators yield similar results when the refinement direction is parallel to the Cartesian axes of the reference frame, at least for the cubic spline interpolation. Still, the central difference method yields a very accurate surface localization in this case. For the center of gravity method, however, only the Sobel operator yields a fairly accurate localization of the surface.

For a point located at 45°, Figure 3.8b shows that only the central difference operator yields accurate results, both for the cubic splines interpolation and for the center of gravity methods.

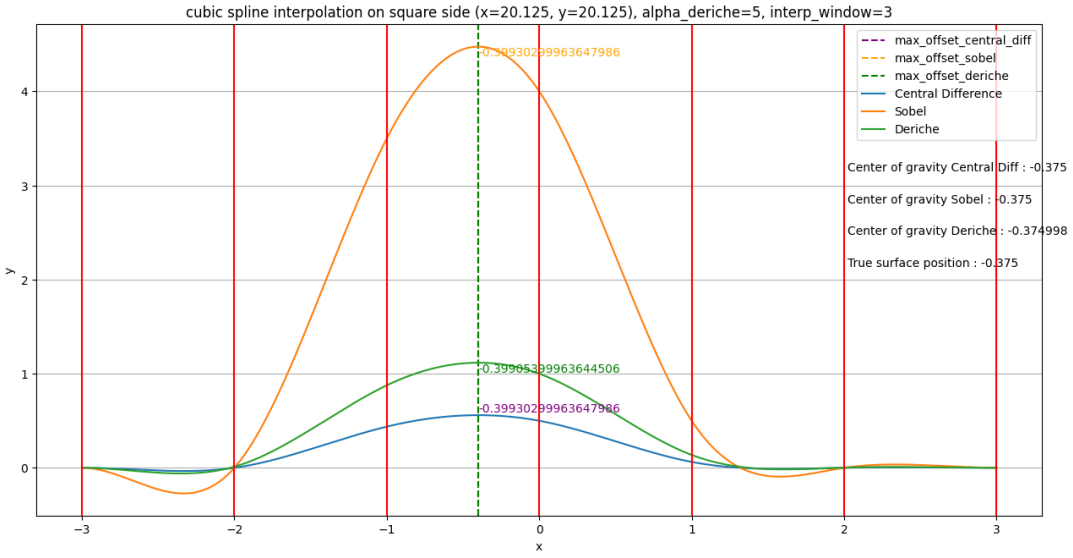
While these results do not explain why some gradient operators work better than the others in certain situations, they allow us to make the practical observation that the central difference operator appears to be the most accurate one in most cases. Therefore, it may be advantageous to use this operator in our implementation. This is also convenient as it is fast and simple to implement.

3.4.2 Cubic Spline Interpolation

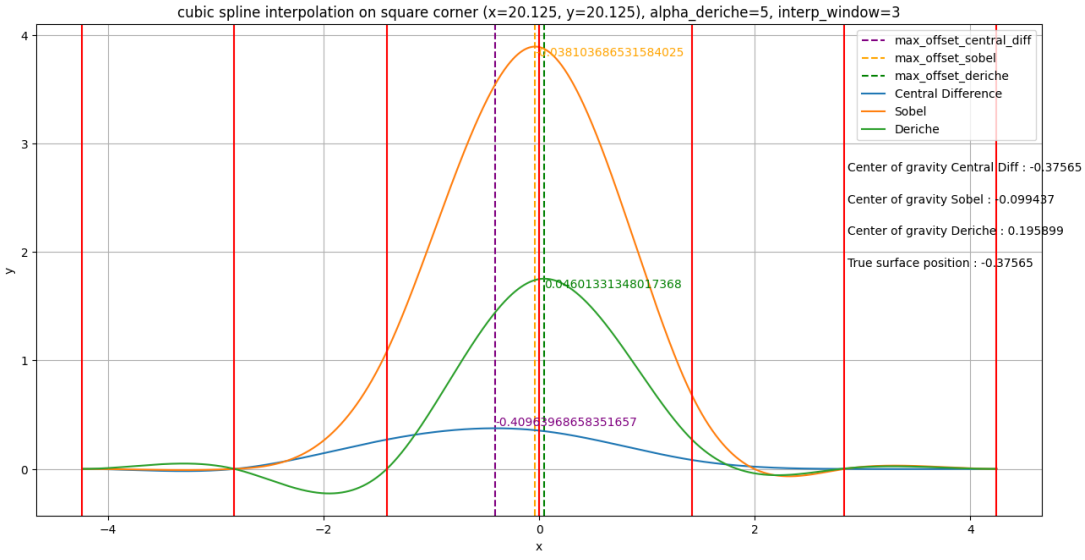
We now decide to investigate the behavior of the cubic spline interpolation while varying some parameters and independently of the gradient operator. To this end, we decide to model a perfect material density as an activation function. In particular, we choose the hyperbolic tangent function $tanh_k(x) = \frac{2}{1-e^{-2kx}} - 1$, where k is a parameter influencing the slope of the function and therefore the "thickness" of its derivative. A graphical representation of the function and its derivative is shown on Figure 3.9.

Using this model to obtain gradient values with various values for k, the slope of the gradient, we create an interpolation model using cubic splines. We still need to introduce the following parameters :

- nb_points: the number of sample points, or rather the number of sampled gradient values, on which to fit the cubic splines.
- h: the distance between two sample points.

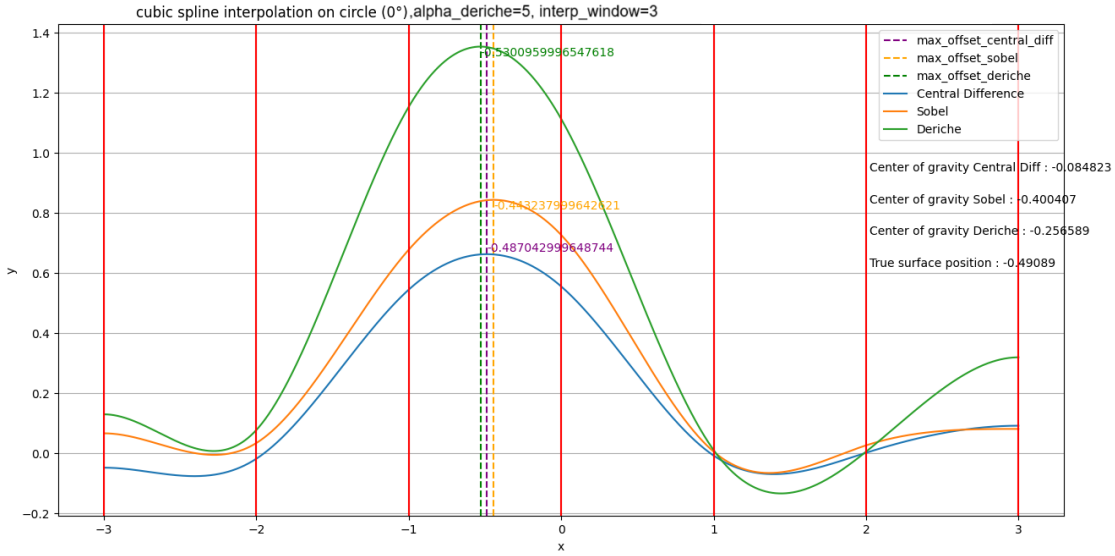


(a) Surface localization results for a point located on the side of the translated 2D square.

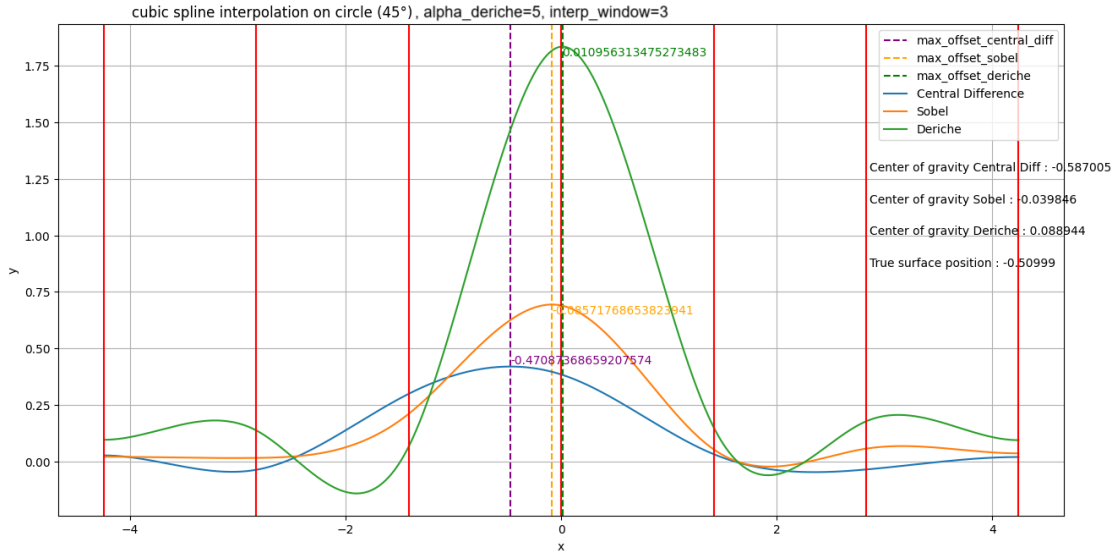


(b) Surface localization results for a point located on one of the corners of the translated 2D square.

Figure 3.7: Graphs of the cubic splines interpolation curves for the different gradient operators, with the surface's localization for each method. The position of the surface computed using the center of gravity method is also displayed on the right for each gradient operator. These results correspond to specific points on a 2D square. The vertical red lines correspond to the localization of the centers of the voxels surrounding the point to be refined.



(a) Surface localization results for a point located at a 0° angle on the surface of the voxelized disk.



(b) Surface localization results for a point located at a 0° angle on the surface of the voxelized disk.

Figure 3.8: Graphs of the cubic splines interpolation curves for the different gradient operators, with the surface's localization for each method. The position of the surface computed using the center of gravity method is also displayed on the right for each gradient operator. These results correspond to specific points on a 2D disk. The vertical red lines correspond to the localization of the centers of the voxels surrounding the point to be refined.

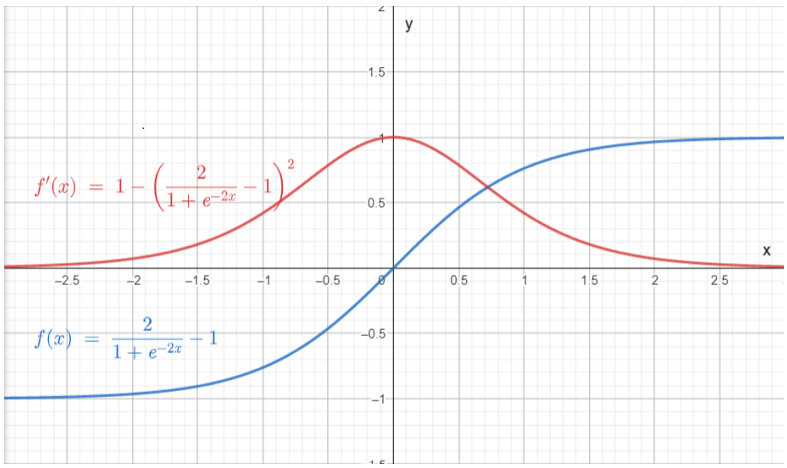


Figure 3.9: Blue: graph of the hyperbolic tangent function. Orange: graph of the derivative of the hyperbolic tangent function.

- ξ: parameter representing how much the hyperbolic tangent is offset horizontally relative to the center of the reference frame. This simulates the distance between the reconstructed surface and the closest voxel's center.

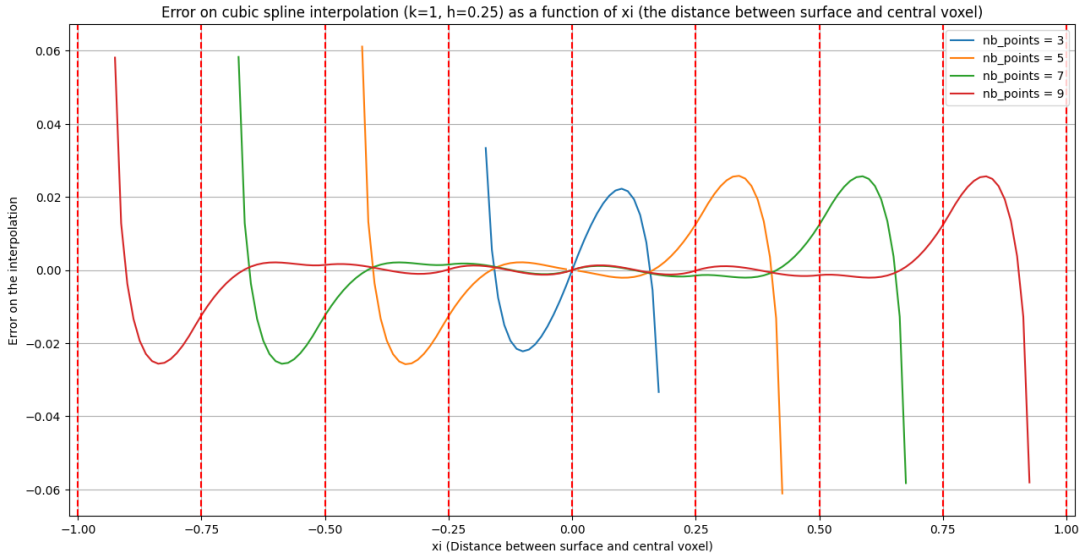
Intuitively, we expect the ratio $\frac{k}{h}$ to control how widely and how densely the peak of the derivative of the hyperbolic tangent function is sampled for the cubic splines interpolation.

In Figure 3.10, we investigate two realistic $\frac{k}{h}$ ratios and plot the error on the predicted surface location as a function of ξ, where ξ represents how far from a voxel's center the true surface location lies.

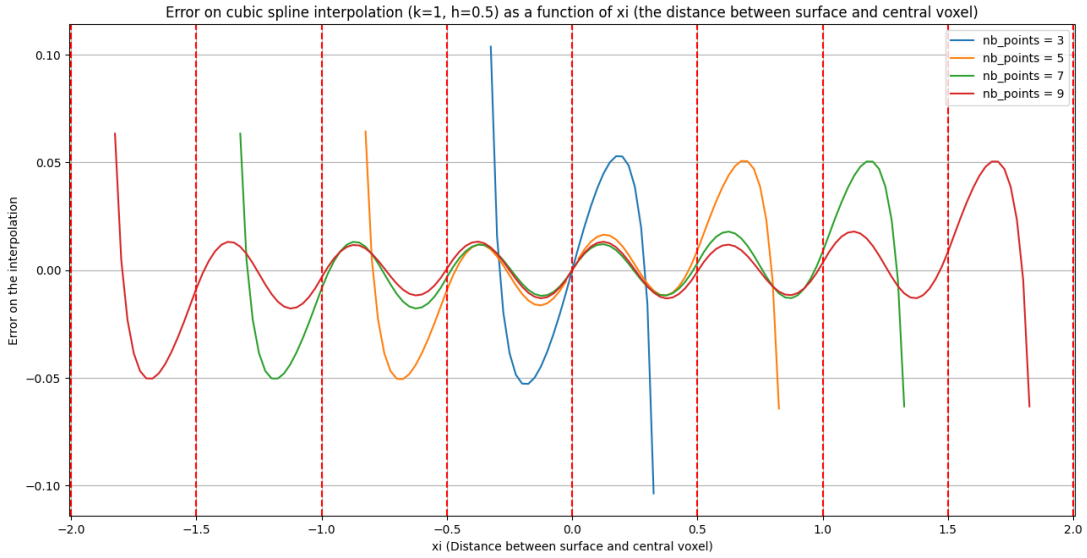
These results demonstrate how important it is to use a sufficiently large interpolation window, as the error explodes very quickly when the true surface lies far from the voxel's center around which the interpolation window is sampled. This behavior is clearly visible on both Figure 3.10a and Figure 3.10b, demonstrating that this particular phenomenon is independent of the $\frac{k}{h}$ ratio.

In a realistic scenario, the voxel centers that are chosen by Canny or any other method to determine a preliminary surface to be further refined should not be more than one voxel away from the true surface location. Indeed, if it were to be the case, it would probably mean that there was a failure at the preliminary surface detection step. Therefore, both Figure 3.10a and Figure 3.10b lead us to think that choosing $nb\_points = 7$ is sufficient to obtain accurate results when ξ is bounded to one voxel.

Finally, one drawback of this method becomes evident when comparing Figure 3.10a and Figure 3.10b. On the first image, where $\frac{k}{h} = 4$, we notice that choosing a bigger interpolation window leads to consistently higher accuracy over a wider span of ξ, because the curves do not oscillate a lot. In contrast, the second figure shows that choosing another $\frac{k}{h}$ ratio leads to oscillations of bigger amplitude and that choosing a bigger interpolation window does guarantee accurate results even for small ξ. In a real CT reconstruction, we cannot exert control over the value of k since it depends on the difference in material density between the object and the air surrounding it, as well as the homogeneity of this density. However, the parameter h can be controlled since it corresponds to the resolution of the image acquisitions. Some tweaking of the resolution could therefore be performed to obtain more consistent accuracy, but it is often fixed by the customer and cannot be chosen at refinement time.



(a) $\frac{k}{h} = 4$



(b) $\frac{k}{h} = 2$

Figure 3.10: Graphs of the error between the true surface position and the position predicted by cubic spline interpolation, as a function of ξ, the distance between the surface location and the closest voxel's center. The gradient is modeled as the derivative of the hyperbolic tangent function. The vertical dashed lines represent the position of the centers of the voxels located within the interpolation window.

3.4.3 Surface Normal Orientation

To determine the direction in which to refine a point's position, we approximate the orientation of the surface normal by constraining it to the main direction that maximizes the gradient magnitude, where main direction refers to one of the 13 main directions of a cube as illustrated on Figure 3.3. This approximation is a source of inaccuracy in the reconstructed surface, and we want to measure its impact. To this end, we will perform some experiments on the 2D voxelized disk we presented in Figure 3.5. In particular, we will apply the sub-voxelic refinement algorithm described in Algorithm 3 to the voxelized disk, with modifications to the choice of surface normal orientation. We will then measure the error on the reconstructed surface as a function of the angle to the center of the disk, but only over one quadrant of the disk since this problem is symmetrical.

To understand the results that follow, we first need to make an important distinction. The 13 main directions that we used in three dimensions become the 4 main directions in two dimensions, which could correspond to the -45°, 0°, 45°, and 90° angles on a trigonometric circle. Among those 4 main directions, we refer to the 0° and 90° angles as the *orthogonal* directions, while the -45° and 45° angles are referred to as the *diagonal* directions.

First, let us measure the error obtained using the 4 main directions approximation, where the main direction that maximizes the gradient magnitude is chosen as an approximation of the surface normal's orientation. Figure 3.11 contains the following elements :

- In green: the signed distance between the true circular surface and the reconstructed surface, which represents the reconstruction error.
- In red: an indicator function which indicates the angles where a *diagonal* direction was determined to maximize the gradient and therefore chosen to be the direction in which to refine the associated surface point.

We notice several surprising results :

1. Very few *diagonal* directions were chosen. In fact, only 4% of the directions were diagonals. Moreover, the angles where a diagonal was chosen seem to be displaying the highest errors.
2. We intuitively expected the error to be minimal around -45°, 0°, 45°, and 90° since these are the locations where the 4 main directions are not approximations as they are equal to the true surface normal's orientation. While the error is indeed minimal for the 0° and 90°directions, the 45° direction displays the highest errors.

To verify if *diagonal* directions are globally performing worse, we first redo the experiment, but this time imposing *diagonal* directions between 22.5° and 67.5°, as this is the range of angles where diagonals are a better approximation of the true normal orientation. The results can be seen on and show that indeed, the *diagonal* directions yield poor results in comparison to *orthogonal* directions.

To check the consistency of these results, we also decided to stop approximating the surface normal and to use the true orientations of the surface's normal instead. One of the purpose of constraining it to 4 main directions was that a voxel's gradient value could be used directly in the calculations as the sampled points all correspond to voxel centers, but using true directions forces us to implement a bilinear interpolation to obtain gradient values of sampled locations that do not correspond to voxel centers anymore. Figure 3.13 shows the result of this experiment, and confirms the tendency of *diagonal* directions to yield worse accuracy, regardless of normal approximations.

Since diagonals seem to consistently yield worse results, we redo the experiment, this time imposing *orthogonal* directions only. The results are displayed on Figure 3.14 and seem much more promising than

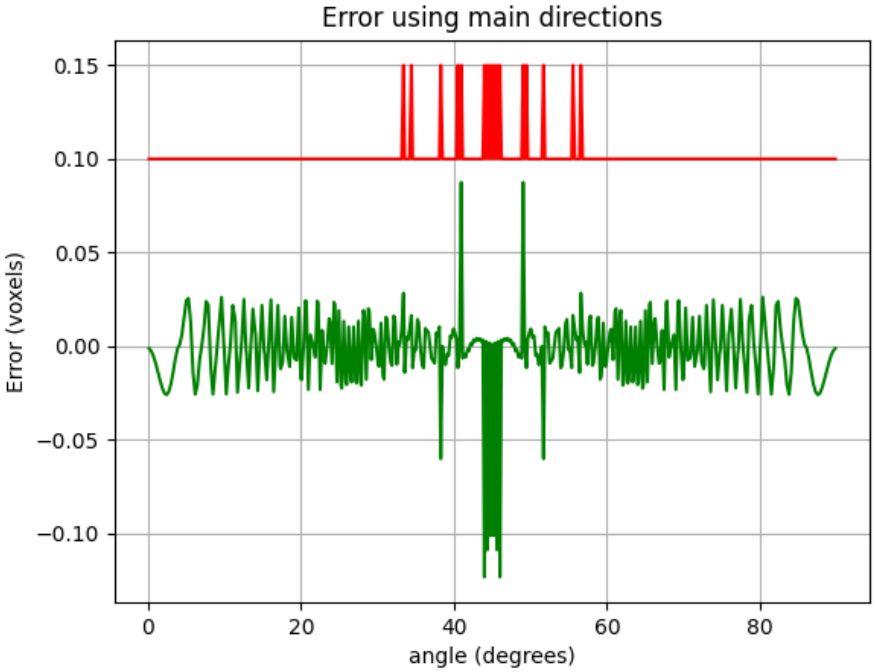


Figure 3.11: Green: error on the reconstructed circle as a function of the angle of the surface points to the center of the 2D voxelized disk. Red: indicator function showing the angles where a diagonal normal orientation was found to maximize the gradient.

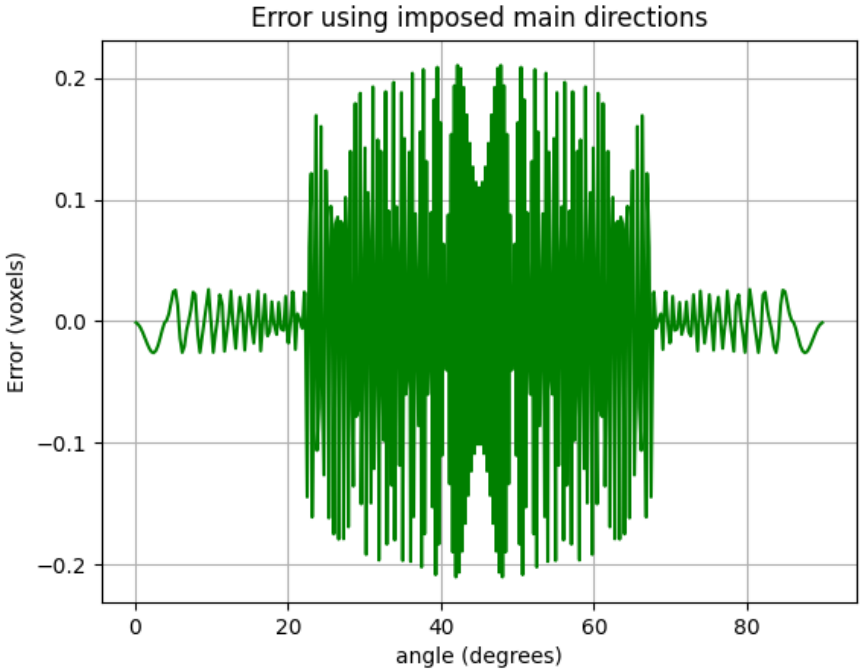


Figure 3.12: Error on the reconstructed circle as a function of the angle of the surface points to the center of the 2D voxelized disk, where *diagonal* directions are imposed between 22.5° and 67.5°.

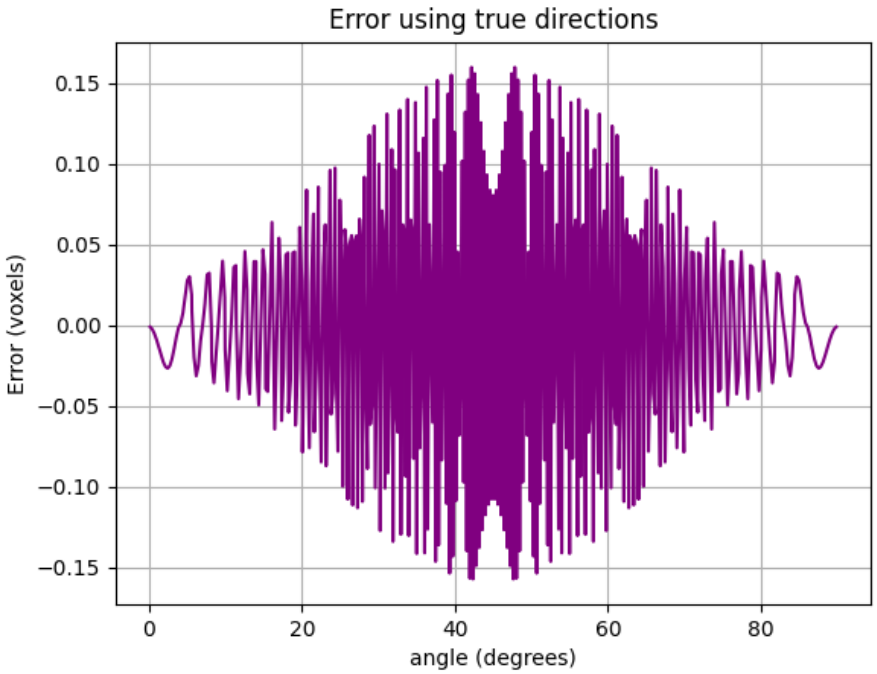


Figure 3.13: Error on the reconstructed circle as a function of the angle of the surface points to the center of the 2D voxelized disk, using the true surface normal orientations.

the results obtained in Figure 3.11. Indeed, the highest unsigned error was 0.14 voxels when we used the *diagonal* directions and is now down to 0.025 voxels, which satisfies the metrology criterion.

In conclusion, it appears that using only *orthogonal* directions leads to better accuracy. While it may have a positive impact if implemented in the 3D method and applied on a real dataset, we are facing trouble explaining the origin of this phenomenon. Indeed, we would intuitively expect this problem to be symmetrical in all directions when working with a circular object.

One possible explanation lies in the way we computed the gradient during these experiments. We used a finite central difference scheme, which involves computing differences of pixel values. Figure 3.15 shows a zoom on some pixels located at the edge of the 2D voxelized disk. When applying this operator in an *orthogonal* direction, the subtraction is performed between pixels that are horizontal or vertical relative to each other. This trajectory is represented in green on the figure. When applying the operator in a *diagonal* direction, the subtraction is performed between pixels that are located diagonally relative to each other. This trajectory is represented in red on the figure. We notice on the figure that the difference in color intensity between diagonal pixels is sharper than the difference in color intensity between horizontal or vertical pixels. This phenomenon is intrinsically linked to the square shape of the pixel, which could be the cause of asymmetry in this problem.

3.4.4 Summary And Results

To conclude on the experiments that were performed in the previous sections, we were able to draw the following conclusions :

- The central difference operator yielded the best results overall, despite little difference with the other operators in most cases.

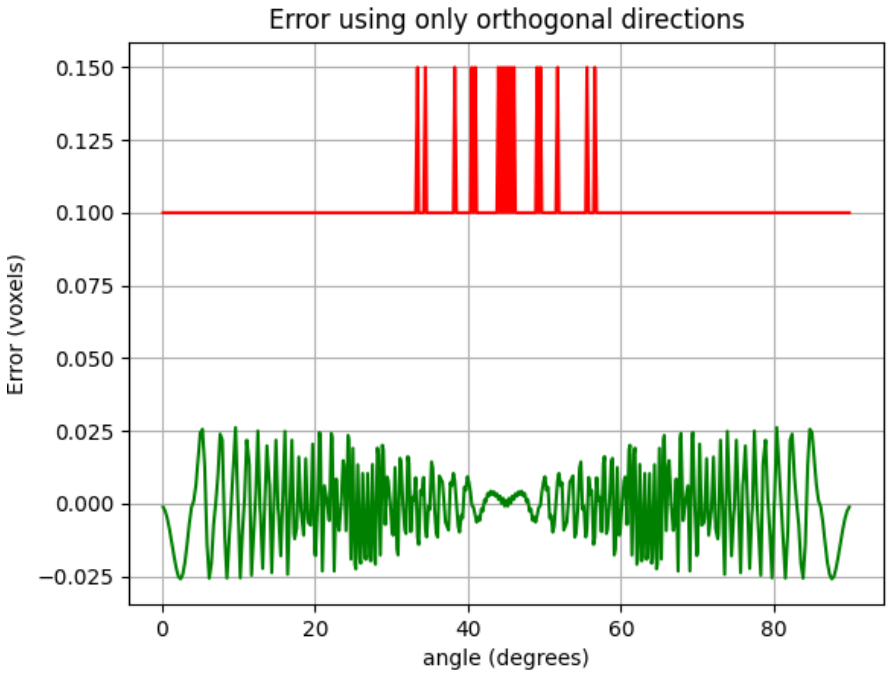


Figure 3.14: Green: error on the reconstructed circle as a function of the angle of the surface points to the center of the 2D voxelized disk, where *orthogonal* directions are imposed everywhere. Red: indicator function showing the angles where a non-orthogonal normal orientation was found to maximize the gradient.

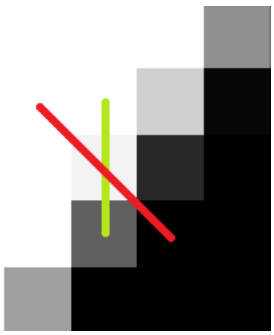


Figure 3.15: Zoom on some pixels located at the edge of the 2D voxelized disk used in the experiments. Lighter voxels correspond to higher density values.

- Using the center of mass of the gradient values to refine the points may yield more accurate results than interpolating gradient values using cubic splines, but the latter technique yields more consistent results.
- Using an interpolation window of at least 7 voxels appears to be optimal.
- Despite being counter-intuitive, approximating surface normal orientations with *orthogonal* directions only seems to yield better results.

Taking these conclusions into account, we perform the sub-voxelic refinement step again with the new design choices. We also compare the resulting error distributions when the quality of the 3D input image data increases. To this end, we simulated new helical CT reconstruction with increasing numbers of image acquisitions involved in the CT reconstruction process. Specifically, the resolution of these reconstructions is 100µm/voxel, and the object performs either 5, 10, or 15 rotations on itself.

Figure 3.16 shows the distribution of the error for each input image, while also comparing the difference between using *diagonal* directions or not.

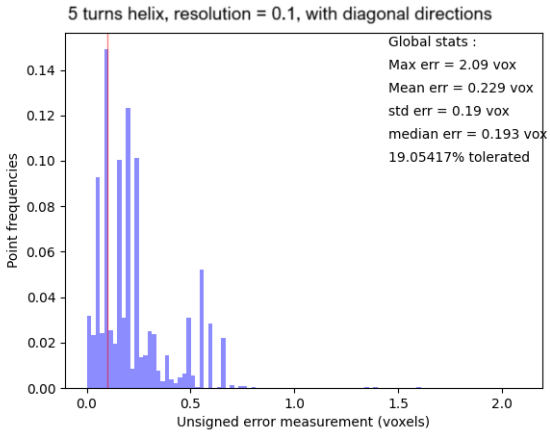
A first observation is that using *orthogonal* directions only does not seem to have a significant impact on the overall accuracy, regardless of the quality of the input data. This result is disappointing as we were expecting it to induce more of a positive impact. It remains interesting not to use *diagonal* directions as it reduces the number of calculations when choosing the direction that maximizes the gradient value.

It is also interesting to notice that the choice of using only a subset of the main directions has a very characteristic effect on the visual quality of the refined point cloud around sharp edges, as points can be pushed away from the tip of the edge. We can consider that the choice of using only a subset of directions enables a trade-off between point density and localization accuracy around such edges. This phenomenon is clearly visible on Figure 3.17, where points can be pushed away from the edges to be more accurately refined, at the cost of a decrease in point density.

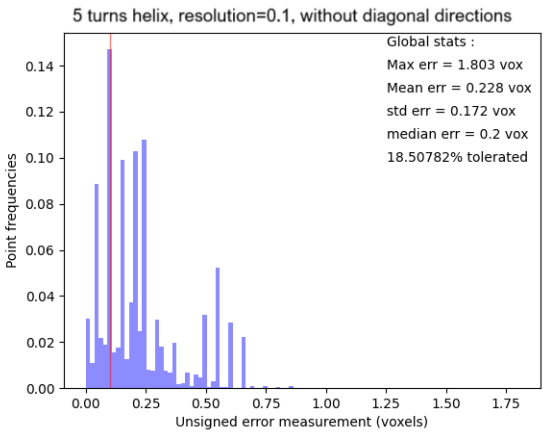
It seems like using an input image obtained via helical geometry with 5 turns is not sufficient to obtain promising results in terms of the quality of the refined points. From 10 rotations and above, however, the quality improves significantly. Indeed, 60% of the points satisfy the metrology criterion, but most importantly, the mean and median errors are below the metrology criterion, which is very promising. There is still a lot of room for improvement, especially as there are still some points which are very badly refined, as the maximum error is over the size of 2 voxels. Moreover, the impact of the quality of CT reconstruction seems to have hit a plateau, as there is no significant improvement in accuracy between images acquired with 10 and 15 rotations. This could be a limiting factor for future improvements.

Figure 3.18a shows the locations of the points that were refined with an error of more than the size of a voxel. It is very visible that these points all belong to sharp edges. Therefore, future efforts could be focused on methods specifically designed for sharp edge reconstruction.

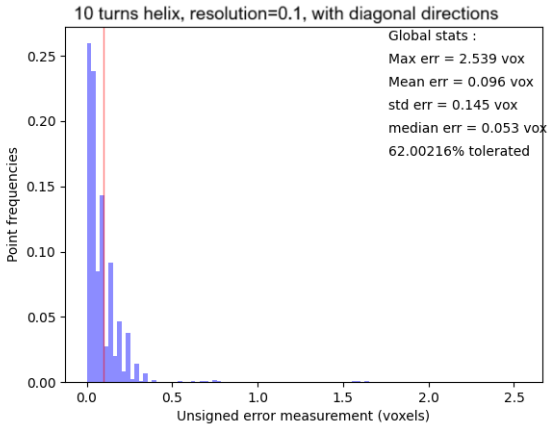
Figure 3.18b shows the locations of the points that were refined with an error of more than one-tenth of the size of a voxel. Said otherwise, these are the locations that do not satisfy the metrology criterion. Similarly to the results we had obtained in chapter 2, the entire bottom of the object is poorly reconstructed. Most of the visible points belong to locations where the input data is more affected by artifacts.



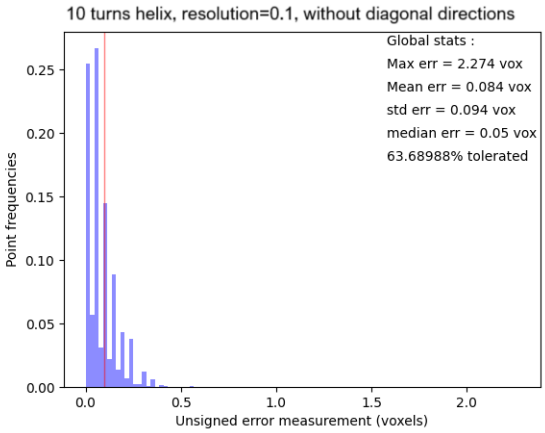
(a) 5 rotations, using diagonal directions



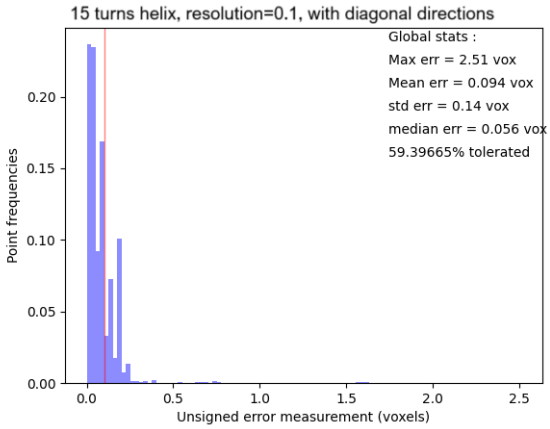
(b) 5 rotations, without using diagonal directions



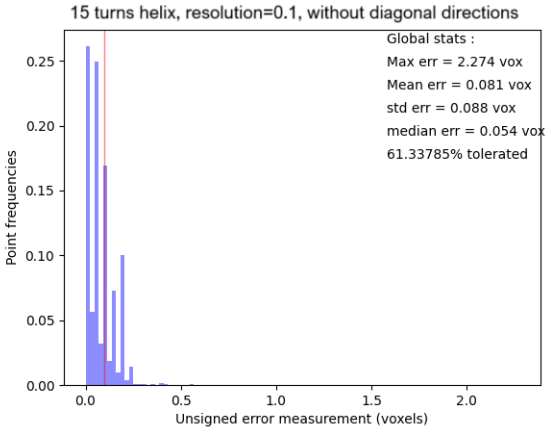
(c) 10 rotations, using diagonal directions



(d) 10 rotations, without using diagonal directions

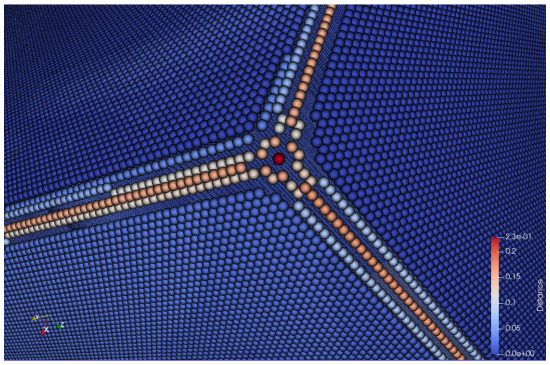


(e) 15 rotations, with using diagonal directions

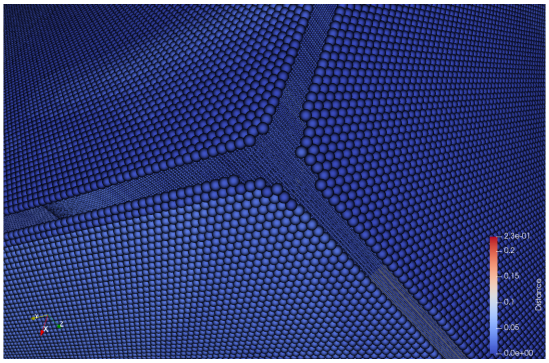


(f) 15 rotations, without using diagonal directions

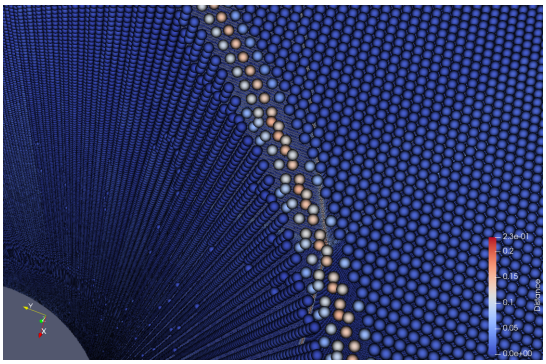
Figure 3.16: Histograms of the unsigned measured error in voxels at every point of the refined point cloud and the reference surface. The vertical red line that is located at x = 0.1 represents the maximal desired error to be under the metrology criterion. Descriptive statistics for the entire set of points and specifically for edges are also displayed.



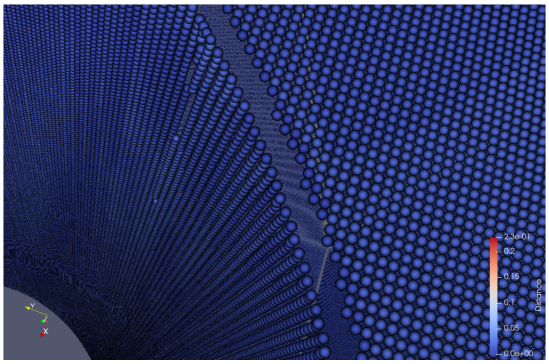
(a) Using diagonal directions



(b) Without using diagonal directions

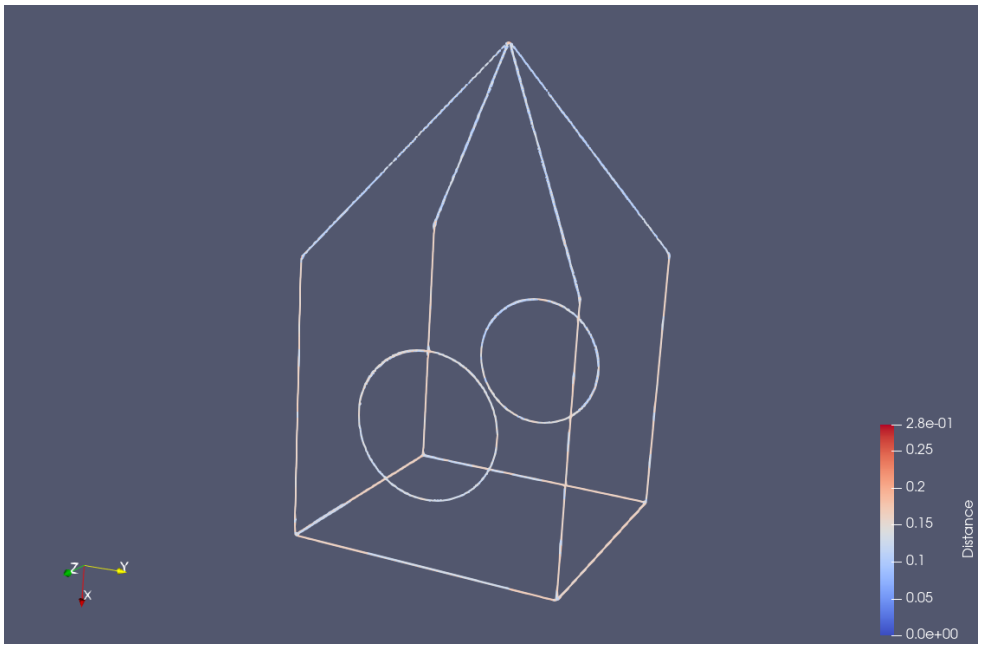


(c) Using diagonal directions

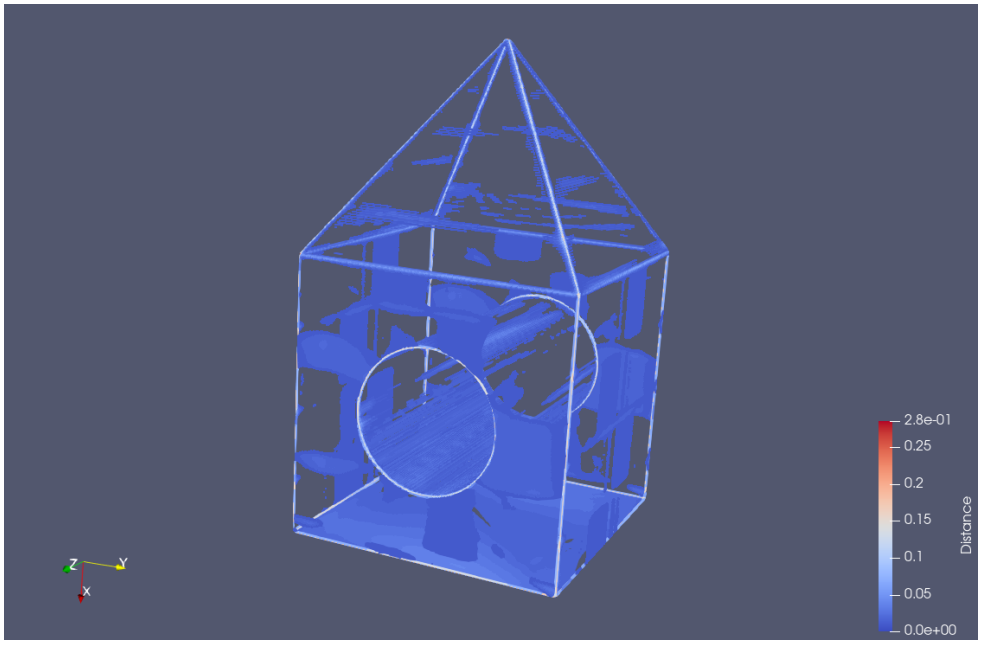


(d) Without using diagonal directions

Figure 3.17: Point clouds obtained after sub-voxelic refinement. The color of the dots indicates the error associated with a reconstructed point. Blue dots represent the lowest error values while red dots represent the highest error values.



(a) Points whose associated deviation is more than a voxel.



(b) Points whose associated deviation is more than the tenth of a voxel.

Figure 3.18: Cloud points with associated error after sub-voxelic refinement on data reconstructed from 10 rotations of image acquisitions.

Conclusion

We started this thesis by reviewing the scientific literature related to the topic of surface extraction for computed tomography. In particular, we found that there are two main families of techniques: one that consists in taking advantage of the information contained in individual CT slices before reconstructing them into a volumetric image, and one that consists in extracting the surface from a volumetric image after it has been reconstructed. The latter family of techniques usually consists in applying a pipeline to first extract a rough estimation of the surface, and then to refine it using some sub-voxelic refinement algorithm. We explored several such algorithms for each step of the pipeline and reviewed their advantages and drawbacks. During this research, we also learned that *VGSTUDIO*, the market leader in CT visualization and CT reconstruction software, established a strict criterion for the quality of extracted surfaces to qualify for metrology applications.

In a second time, we assessed the performance of the surface extraction algorithms that we discovered during our research and that were already implemented inside an open-source software library called *Visual Toolkit*. To conduct this evaluation, we compared the extracted surfaces to a reference surface and obtained point-to-point distances between the two surfaces. We then compared these distances to our metrology criterion to determine whether these algorithms performed sufficiently well on their own to qualify for metrology applications.

We performed these experiments on data acquired via both circular and helical geometries, with the hope that working with input data of better quality would improve the quality of the extracted surface. While we found that helical geometry does indeed yield better results than circular geometry, the reconstructed surfaces were still far from satisfying the metrology criterion.

We dedicated Chapter 3 to the study and implementation of two sub-voxelic refinement methods. The first technique consisted in averaging neighboring pixel values of the input image to refine the position of a point belonging to the extracted surface, in a *center of mass* fashion. This method did not yield promising results, and we decided not to explore it further. The second technique consisted in using the gradient of the pixel values of the input image as an indicator for the refinement of the points belonging to the extracted surface. This technique did not yield significantly better results than the algorithms already implemented in *Visual Toolkit* we had tested previously. We found, however, that this technique displayed some potential for improvement. As such, we dedicated the next sections to the study of some of the design choices of this sub-voxelic refinement method. In particular, we studied the method used for computing the gradient of the pixel values, the method and parameters used for interpolating the gradient values, and the method used for approximating the orientation of the surface normal. This study gave us a deeper understanding of the phenomena underlying these results.

This is an important contribution, as we did not find any scientific paper detailing all the design choices of their pipeline, which no longer feels like a black-box following our study of the different building blocks that constitute the entire surface extraction pipeline. For a company such as X-RIS, which aims at producing its own solutions from start to finish, from the manufacture of CT equipment to the development of its software, it is crucial to obtain a deep understanding of the pipeline in order to be able to use these results in new situations.

Using the combination of parameters deduced from our parameter study, we performed the experiment again on input images of increasing reconstruction quality, to finally extract a surface where 60% of the points belonging to the surface satisfy the metrology criterion, and where the mean error satisfies the metrology criterion as well.

While we did not manage to implement a method that extracts a surface where 100% of the points satisfy the metrology criterion, our method may still be used for other use cases, such as air bubble detection. Moreover, we showed that the largest errors are located at the sharp edges and at the corners of the object, which opens clear opportunities for further investigations

Possible Improvements

While it is clear that using pre-implemented surface extraction techniques does not yield satisfactory results, we think that further improvements to the sub-voxel refinement technique are possible and should be explored. In particular, one could differentiate the method used to extract planar faces from the method used to extract edges. Making this distinction could also be useful to tune the density of points used to reconstruct the surface in the following way: decimate the point cloud belonging to flat surfaces and increase the density of points in edges and corners, allowing for improved accuracy and faster extraction.

Another venue for improvement is to test the algorithm with more realistic sizes of objects. Indeed, our sample object was small compared to real industrial parts, and the execution time could exceed what we consider a satisfactory computation time.

We also did not investigate the behavior of our methods for objects composed of different materials.

Finally, investigating the sign of the error might show that there is a systematically positive or negative error, especially around edges, meaning that applying a corrective factor is a potential solution.

Appendices

Specifications Of The Test Machine

All tests were performed on a single machine with the following specifications :

- CPU: i9 1490KF
- RAM: 128 GB DDR5 6000 MHz
- GPU: RTX 4090 24GB VRAM

Bibliography

[1] North Star Imaging, Inc., "Manufacturing case studies," 2024, accessed: 2025-05-25. [Online]. Available: https://4nsi.com/case-study-category/manufacturing/

[2] Rigaku Corporation. (2022) What is beam hardening in ct? Accessed: 2025-05-21. [Online]. Available: https://rigaku.com/products/imaging-ndt/x-ray-ct/learning/blog/what-is-beam-hardening-in-ct

[3] K. Taguchi, G. Zeng, and G. Gullberg, "Cone-beam image reconstruction using spherical harmonics," *Physics in Medicine and Biology - PHYS MED BIOL*, vol. 46, 06 2001.

[4] G. R., "Master's thesis : Visual tools for computed tomography volume representation: Large data visualisation and surface extraction," 2024.

[5] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, p. 163–169, Aug. 1987. [Online]. Available: https://doi.org/10.1145/37402.37422

[6] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," *Computers Graphics*, vol. 30, no. 5, pp. 854–879, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849306001336

[7] T. Shen, J. Munkberg, J. Hasselgren, K. Yin, Z. Wang, W. Chen, Z. Gojcic, S. Fidler, N. Sharp, and J. Gao, "Flexible isosurface extraction for gradient-based mesh optimization," *ACM Trans. Graph.*, vol. 42, no. 4, Jul. 2023. [Online]. Available: https://doi.org/10.1145/3592430

[8] S. Ontiveros-Zepeda, J. A. Yagüe-Fabra, R. Jiménez, and F. J. Brosed Dueso, "A comparative of 3d surface extraction methods for potential metrology applications," in *Advances in Manufacturing Systems*, ser. Key Engineering Materials, vol. 615. Trans Tech Publications Ltd, 7 2014, pp. 15–21.

[9] M. Fisher, "Marching cubes," 2014, accessed: 2025-05-25. [Online]. Available: https://graphics.stanford.edu/~mdfisher/MarchingCubes.html

[10] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, ser. SGP '06. Goslar, DEU: Eurographics Association, 2006, p. 61–70.

[11] B. Curless and M. Levoy, *A Volumetric Method for Building Complex Models from Range Images*, 1st ed. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3596711.3596726

[12] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, and W. Stuet-zle, "Surface reconstruction from unorganized point clouds," 01 1992.

[13] W. Schroeder, R. Maynard, and B. Geveci, "Flying edges: A high-performance scalable isocontouring algorithm," 10 2015.

[14] W. Schroeder, S. Tsalikis, M. Halle, and S. Frisken, "A high-performance surfacenets discrete isocontouring algorithm," 2024. [Online]. Available: https://arxiv.org/abs/2401.14906

[15] N. Amenta, S. Choi, and R. Kolluri, "The power crust," *Computatinal Geometry*, vol. 19, 03 2004.

[16] N. Amenta, S. Choi, and R. K. Kolluri, "The power crust, unions of balls, and the medial axis transform," *Computational Geometry*, vol. 19, no. 2, pp. 127–153, 2001, combinatorial Curves and Surfaces. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925772101000177

[17] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[18] S. Ontiveros, R. Jimenez Pacheco, J. Yagüe-Fabra, and M. Torralba, "Analysis of surface extraction methods based on gradient operators for computed tomography in metrology applications," *Materials*, vol. 11, p. 1461, 08 2018.

[19] Y. Nagai, Ohtake and H. Suzuki, "Accurate surface extraction on ct volume using gradients obtained by differentiating fdk formula," *e-Journal of Nondestructive Testing*, vol. 24, 03 2019.

[20] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/026288569290066C

[21] F. Banterle, M. Corsini, P. Cignoni, and R. Scopigno, "A low-memory, straightforward and fast bilateral filter through subsampling in spatial domain," *Comput. Graph. Forum*, vol. 31, no. 1, p. 19–32, Feb. 2012. [Online]. Available: https://doi.org/10.1111/j.1467-8659.2011.02078.x

[22] I. Sobel, "An isotropic 3x3 image gradient operator," *Presentation at Stanford A.I. Project 1968*, 02 2014.