



# Graphical Methods For Online Surface Fitting On 3D Range Sensor Point Clouds

by

Daniel Cernea

a thesis for conferral of a Master of Science in Computer Science

Supervisors:  
Prof. Dr. Lars Linsen  
Prof. Dr. Andreas Birk

August 24, 2009 - Bremen

---

School of Engineering and Science



---

# Contents

---

<b>Contents</b>	<b>i</b>
<b>1 Introduction</b>	<b>3</b>
<b>2 Problem Description and Analysis</b>	<b>7</b>
2.1 Noise Reduction on 3D Range Sensor Data . . . . .	11
2.2 Surface Reconstruction Through 3D Mesh Merging . . . . .	25
2.3 Mesh Simplification and Multiresolution Representation . . . . .	33
2.4 Surface Uncertainty Visualization . . . . .	39
<b>3 Real-Time Graphical Methods for 3D Point Cloud Processing</b>	<b>49</b>
3.1 Error Reduction with Anisotropic Diffusion and Wavelet Denoising	52
3.2 Surface Multiresolution Representation . . . . .	65
3.3 Mesh Matching and Merging . . . . .	70
3.4 Uncertainty Visualization Methods for Surfaces . . . . .	72
<b>4 Conclusions and Discussion</b>	<b>79</b>
<b>5 Future Work</b>	<b>81</b>
<b>Bibliography</b>	<b>85</b>
<b>A Terminology</b>	<b>93</b>
<b>List of Symbols and Abbreviations</b>	<b>95</b>
<b>List of Figures</b>	<b>96</b>



---

# Abstract

---

Point clouds are one of the most common data structures obtained with current day technology for scanning real-life objects or various environments. As such, the importance of digitally reconstructing the initial surfaces from the gathered boundary points is vital in many scientific fields. Our focus goes towards the reconstruction of environmental information from 3D range sensor point clouds to facilitate a more accurate description of the surroundings. One major application of this comes to life in the area of robotics, where a precise environment description can facilitate robot and object localization, area mapping (SLAM), feature detection and many more.

We discuss how we can use the particularities of such a 3D range point cloud in order to reduce the noise levels introduced by the scanning sensors. Further we present possibilities for visualizing the uncertainty of the reconstructed surfaces from the denoised point cloud data, as well as different methods that would allow for a multiresolution 3D representation of the scanned environment.



## Chapter 1

---

# Introduction

---

For years, modern sensors enabled us to explore features of the real world that our five human senses could not perceive. These sensors and the corresponding information has an even higher importance in a world of powerful computational devices that allow the proficient use of this data in fields like manufacturing, geography, architecture, design, medicine and many more.



Figure 1.1: 3D point cloud rendering example from a music video

A large part of these sensors are meant to gather information about objects or environments where the human (mainly visual) capabilities would be useless or insufficient. Some of the new abilities we have gained this way include seeing without the presence of light, exploring density, consistency and other object characteristics, as well as mapping inner and outer boundaries of materials.

The last mentioned area of structural digitalization of real-life objects is a widespread

topic in science and industry, as it enables in-detail scans of surfaces and their virtual representation and modification. Tasks involving 3D scans of objects can be used for developing different virtual reconstructions of surfaces for later use as prototypes, as well as for alternative methods for video recording by substitution of video cameras with 3D range sensors that can result in artistic visualizations\* through point cloud representations (Figure 1.1).

In this paper, our focus goes towards point cloud information as 3D digitized data defining an entire object or environment, or simply a subset of it. The structure of the cloud consists of a three-dimensional set of points obtained with some type of measurement device that scans the surface of a material object. A representation of a 3D range point cloud is presented in Figure 1.2, where the  $x$  and  $y$  axis of the initial 3D space are mapped to the two dimensions of the image, while the depth  $z$  of the generated points is normalized and displayed in grayscale. Once a point cloud is computed, it can later be processed by an algorithm that would group points and fit these groups to a 2D or 3D shape, with a position and orientation in the 3D coordinate system.

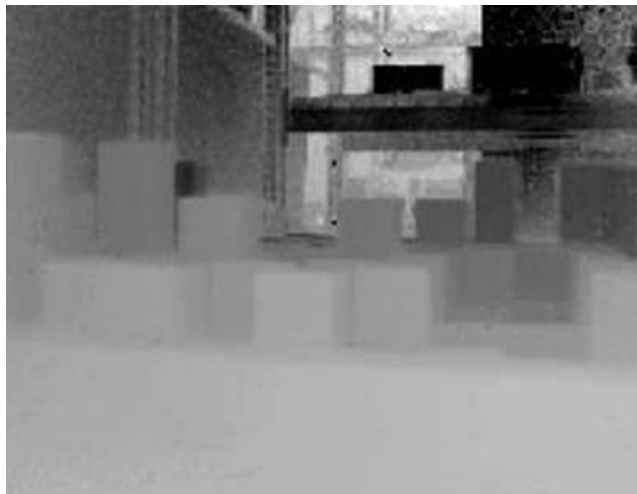


Figure 1.2: Grayscale image of the normalized distance values generated by a TOF sensor

While the current chapter is an introduction to the topic at hand, in Chapter 2 we present recent research in visualizing and reconstructing patches or complex surfaces from 3D point clouds, as well as the proposed research of the thesis, organized in four subsections. The first subsection analyzes the denoising methods that can be applied to the point cloud data in order to obtain a more accurate distance measurement.

---

\*Figure 1.1 - Music video by Radiohead - House Of Cards, available as public stream at <http://www.youtube.com/watch?v=8nTFjVm9sTQ>



In Sections 2.2 and 2.3 we consider different surface representations that allow for a proper merging of point cloud based meshes and the customization of the level of detail (LOD), thus enabling computation complexity management. The fourth subsection focuses on the visualization of uncertainty in the digitalized surfaces. In the end, we point out the main features of the desired implementation.

The goal of this research is to implement graphical and visualization methods that support a fast and reliable digitalization of 3D environments without human intervention, with the secondary objective of combining robotics and graphics in order to give new solutions to SLAM and other similar problems.



## Chapter 2

---

# Problem Description and Analysis

---

In this chapter we present the directions we would like to follow in our research on fitting surfaces or patches to 3D range point clouds, as well as reasons for their viability for the proposed goals, based on related approaches and publications.

The discussed field of fitting various surfaces to point cloud data is vast and incorporates many problems. These issues need to be handled as efficiently as possible, in order obtain algorithms capable of solving the major task of automatically fitting surface representations to sensor information, structured in the form of point clouds. The complexity of the problem increases even more when all the issues have to be solved in an online fashion. As an example from surface reconstruction, this would translate not only into correctly detecting important gaps (Figure 2.12) in the scanned areas, but also doing this in a limited amount of time, or from a computational perspective, in a limited number of processing steps.

One of our goals is to accurately reconstruct 3D objects and environments. This process can be directly coupled with the sensor data that we are using for reconstruction. Thus, we would like to avoid using artificially generated datasets of 3D scans, as the results of the algorithms we would implement and apply rely also on the noise distribution present in the scans. As different sensors have different types of noise distributions, we decided to use real world datasets that have the advantage of testing the proposed methods with the conviction that the results will be representative for real scenarios.

A particular type of device that we can use to acquire point cloud data is represented by the time-of-flight (TOF) sensor. Some of these devices have additionally the advantage of generating 3D range images, with the points being distributed in a Cartesian coordinate system. Under these circumstances, we can make use of this

property that the sensor scans generate points at a regular interval on two axes, thus mapping the information to a 2D grid.

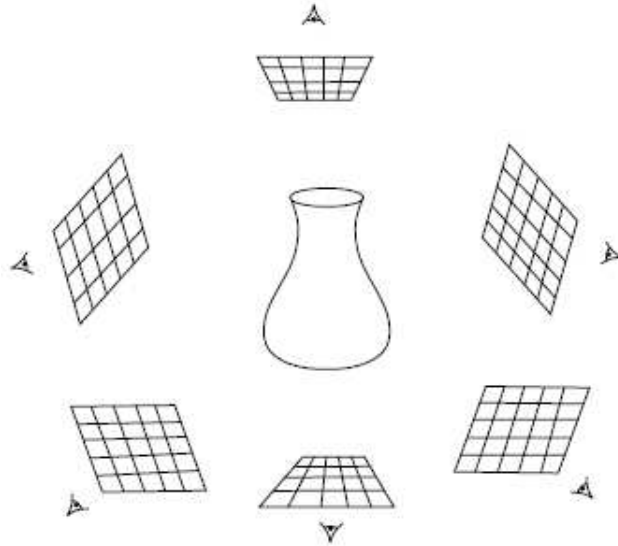


Figure 2.1: Range images of an object taken with the range sensor orthogonal to the 6 surfaces of a cube that bound the object

In other words, when a single scan is displayed in a virtual space with the viewpoint positioned and oriented as the sensor at the recording moment, we would obtain a grid of points similar to that of pixels on a screen (Figure 2.1). In the same way as for pixels, we would have the properties that two horizontally or vertically consecutive points having a constant distance between them and no two points are ever positioned on the same line that passes through the viewpoint.

The input data for our research will be obtained from sensors widely used in academic research and industrial products for generating point clouds from surface scans (with the purposes presented in Chapter 1), namely the Swiss-ranger (SR)[12] and the rotating laser-range-finder (LRF)[24]. While both these sensors are TOF devices, there are fine differences between them: resolution, point cloud structure, scan time etc.

Our work will focus on using point clouds generated by a Swiss-ranger mounted on top of a mobile robot. These TOF sensors offer ideal conditions for achieving our goals, by generating data with a lower range image resolution - faster frame processing, small scanning intervals - forces the need for online solutions, and reduced view angle - which allows for a better mapping to 2D. The dataset was generated by the Robotics Department of Jacobs University and contains multiple groups of scans,

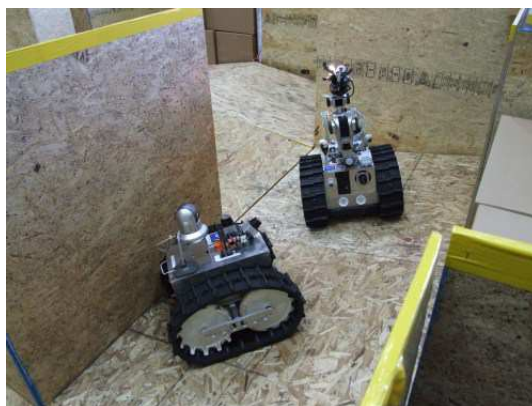


Figure 2.2: Two Rugbots (rugged robot) in a rescue scenario with an environment similar to the Jacobs University Rescue Arena. The robots are orienting themselves by using mainly range images generated by the sensors mounted on them. The recorded range images are locally processed, but also forwarded and stored to a database.

with a number of 100 to 2000 consecutive frames of an in-door environment in Jacobs University's on-campus premises, the Jacobs University Rescue Arena (Figure 2.2). Each frame contains 25.344 points, as the resolution of a Swiss-ranger generated depth image is 176x144 pixels. The points are defined by their position in the 3D coordinate system of the sensor  $(x, y, z)$  and an associated intensity  $I$ . Technical specifications of the device that are important for the thesis also include the facts that the distance resolution is 1% of the range and the typical frame rate is 25 fps, with variations between 15 and 30 Hz.

The second dataset was generated by an LRF at the University of Osnabrück, Germany [7]. It represents an out-door range recording at the AVZ building of the mentioned university, and contains 63 frames of 81.360 data point each. The position of each scanned point is stored as previously described.

For both datasets we have access to the odometry information of the mobile robots, deduced from gyroscopes and represented by a set of values in global coordinates  $(x, y, \theta)$  given for each frame. Nevertheless, odometry data on many mobile robots can be strongly imprecise and noise prone due to the slipping of the wheels or lack of engine precision.

Another advantage of these datasets is that they have been previously used for SLAM (Chapter A)[7][46] by different academic communities, therefore supplying us with the possibility to compare our surface reconstruction efforts, analytically and visually.

In the following subsections we concentrate on four main topics of proposed research,

presenting the ideas behind them, as well as particular work executed on tasks with similar goals or similar implementation methods, in order to point out procedures we would like to extend or elements we would like to adapt for the topic at hand.

## 2.1 Noise Reduction on 3D Range Sensor Data

Digital images are nowadays present in every aspect of the human existence, be it private or professional. But this new era of imaging has also eliminated the limitation of representations to the visible spectrum. Nowadays visualizations allow us to gather insight into the world of medicine, physics and chemistry in ways never imagined before.

As these images that plot distances, directions and many other features into colors and shapes gain increasingly more ground as the basis of various analytical procedures that extract vital information from them, the importance of having an accurate representation of the data is positioned in the center of researchers attention.

Such methods that modify an image, be it natural or artificial, in order to improve the results obtained by applying specific algorithms on the data later on, are called *preprocessing methods*. A particular type of preprocessing methods that have steadily gained importance over the last decades is represented by the noise reduction preprocessing.

Visualized data is prone to multiple types of errors, either depending on the acquired data, or on the processing and visualization step (Figure 2.15). As their name suggests, preprocessing methods are executed on a dataset before it is used by a specific processing algorithm, aimed towards reaching different types of conclusions or extracting information from it. As such, the only level of error they have access to, and thus eliminate, is the acquisition noise usually introduced by the sensor device that generated the data.

A particular set of information that can be visualized as 2D representations is the 3D range sensor data. These datasets have the characteristic that, for particular types of TOF sensors, the information is obtained as a 2D grid of distances and intensities. By using normalization of the intensity values, but more importantly of the distances, a 2D grayscale image can be generated representing the previously mentioned grid. The color of each pixel in the grid is obtained by the normalized mapping of the scanned distance to the color space (0–255 values for a color encoding of 1 byte per channel).

A 3D range sensor image generated in such manner allows for a much simpler analysis of the point set in a 2D image-space, thus reducing the processing complexity and additionally offering the possibility of applying graphical methods specifically designed for 2D images, in order to enhance or extract robust feature descriptors.

Nevertheless, sometimes the most important features that decide over the success of the processing step are deteriorated by the introduced acquisition noise. This is

especially true in the case of range images obtained from infrared (IR) sensors, which can generate highly noisy data because of spurious readings influenced by elements like additional IR sources (e.g. ambient light), surface properties of the scanned objects (dark objects tend to absorb IR light, offering a minimal reflection to the sensor, thus influencing the quality of the reported information) and cut-off distances (TOF sensors usually have a maximal distance range, which if overstepped will lead to a reading equal to the threshold value). As our sensors are not perfect and we can not change the environment such that the obtained information completely lacks errors, the only suitable path to follow is the introduction of preprocessing steps that reduce the influence of noise on the dataset.

An important step towards achieving this is represented by the analysis of the type of errors hidden in the data. According to [14], 3D range images present the highest level of noise along the primary directions of the sensor's lines of sight. This is of particular importance when dealing with range images, as we know that the vertical and horizontal scanning errors, or in other words the inconsistencies in the spacing of the grid, are minimal and can be disregarded. Additionally, it is proven that the noise introduced by the scanning process along the  $Z$ -direction has an asymmetric error distributions [14], knowledge that is valuable for any algorithm trying to eliminate the inherent noise.

Measurement errors do not represent the only problem with range sensor point clouds. A very important characteristic of these point sets, that is partially considered in this section and further emphasized in the next ones, is the amount of generated points. With large 3D point datasets, our goal is not only the elimination of noise or filtering of imperfect data, but also redundancy reduction [38] implemented by down-sampling the point set. This complexity reduction can be applied homogeneously or selectively, depending on the detail levels of the scanned surface.

In general, error reduction on any given point cloud is executed in 3D space [38] as this represents the naturally generated data. One method that is used for this purpose presents a discrete Laplacian smoothness operator that moves one point  $P$  to the centroid computed by  $\frac{1}{k} \sum_{i=1}^k q_i$  of the 1-ring formed by the points  $q_1, \dots, q_k$ . Similar algorithms can be applied on 3D point clouds by replacing the 1-ring by the 3D neighborhood of the point  $P$ .

However, the Laplacian smoothness operators is known to cause the shrinkage of the entire mesh, fact extremely undesirable for visual information. Even if other algorithms have overcome this limitation, the computational expense of reducing or even completely eliminating noise in a three dimensional environment is still high.



What we propose in this section is an alternative that considers - where possible - the 3D point cloud as a range sensor image and applies modifications and combinations of specific image denoising algorithms to lower the error levels in the data.

The main idea is to reduce the noise attached to the attributes that are used in the main processing step, and not necessarily to reduce the error levels in all attributes. But reducing the amount of erroneous information in images is particularly improved by the usage of normals, especially when detecting and reconstructing surfaces [29]. Therefore, generating and reducing the errors in surface normals is of high importance for our endeavor. Likewise, when we build surfaces from measured data, we would like to reduce the effects of noise on visualization or subsequent processing.

The computation of normals specific to each range point is not a trivial task. Multiple methods can generally be applied for these purposes, as presented in [29] and [40]. The main idea behind the algorithms is represented by the application of the total least square method to the  $k$  nearest neighbors of the point  $P$  [62], in order to generate a local surface. By using this surface, the normal at point  $P$  can be approximated as the normal of the obtained surface at that position.

In our implementation of the least square method we have to consider that the accuracy of the normal estimation depends not only on the noise levels and the neighborhood size  $k$ , but also on the curvature of the surface and the distribution of the samples. Nevertheless, in our case the datasets considered contain mainly planar surfaces and the distribution of the samples is homogeneous in 2D space.

As the actual surface around a point  $P$  can have a variable complexity, the method of manually setting the value of  $k$  [29] can be significantly improved as presented in [40]. This approach enables us to compute the neighborhood size relatively to each point, based on the local topology, noise and sampling density, which together with a previous error filtering executed on the point cloud, suggest a high level of accuracy for the computed normals.

As the underlying points for the estimation of the local surface are affected by noise, one should for all normal extraction methods first filter out the noise of the range image by a smoothing filter. Still, these algorithms influence and even eliminate edges, an undesired result for our images. Particular methods for smoothing 2D images without influencing the edges will be presented later in this section. By reducing the noise inside recorded surfaces and maintaining the surface manifolds, the chances of computing close to correct normals should be strongly improved.

Once the normal values are computed for the 3D range image, we can use this

information together with other attributes (i.e. distance) with the purpose of improving again on the overall precision of the depth images. To achieve this image filtering with minimal effects upon the edges of the objects, we will explore variations of two widely used filtering methods: denoising methods and anisotropic diffusion. Additionally to these, other filtering methods might be considered during our research.

The denoising of images is a process that can typically be described based on the following chain of events:

- the image is transformed into a different N-dimensional (ND) space ( $N=1 \rightarrow \infty$ ) that allows for a better detection of the noise information
- a threshold is applied on the data inside the ND space, in order to remove the detected error levels
- the inverse of the initial transformation is applied and the information is represented, denoised, in the original image space

A valid possibility for the ND space, which has already presented promising results for our denoising tests, is represented by the domain obtained after computing  $N \times D$  ( $N$  - normal,  $D$  - distance) for each point in the point cloud  $C$ . As  $N$  can be decomposed into 3 elements, which are basically the projections of the vector onto the axes, we obtain a 3D space based on:

$$x' = N_x \cdot D, y' = N_y \cdot D, z' = N_z \cdot D, X' = (x', y', z') \quad (2.1)$$

Where  $X'$  is the corresponding point in the new point cloud  $C'$ , for the point  $X$  from  $C$ .

In this new domain, points with similar distances and similar normal orientations will be grouped together in areas of the 3D space. But distance and normal similarity is a defining property for points that belong to the same planar surface. Thus, the clusters in the 3D point cloud  $C'$  can be considered as surface correspondences and interpreted as a mixture of Gaussians, where each Gaussian represents a real quasi-planar surface (Figure 2.3).

An additional possible domain mapping is given by the Fourier transformation, which follow the same basic idea:

$$\text{Spatial domain } h(x, y) \Rightarrow DFT \Rightarrow \text{Transformation domain } F(u, v) \quad (2.2)$$

Here, the ND space, also called the *frequency domain*, has the distinct advantage of detecting the presence of frequencies in the original image. But due to the fact that, as previously mentioned, noise inside a TOF sensor acquired datasets has a asymmetric error distribution, we could have the possibility of eliminating it.

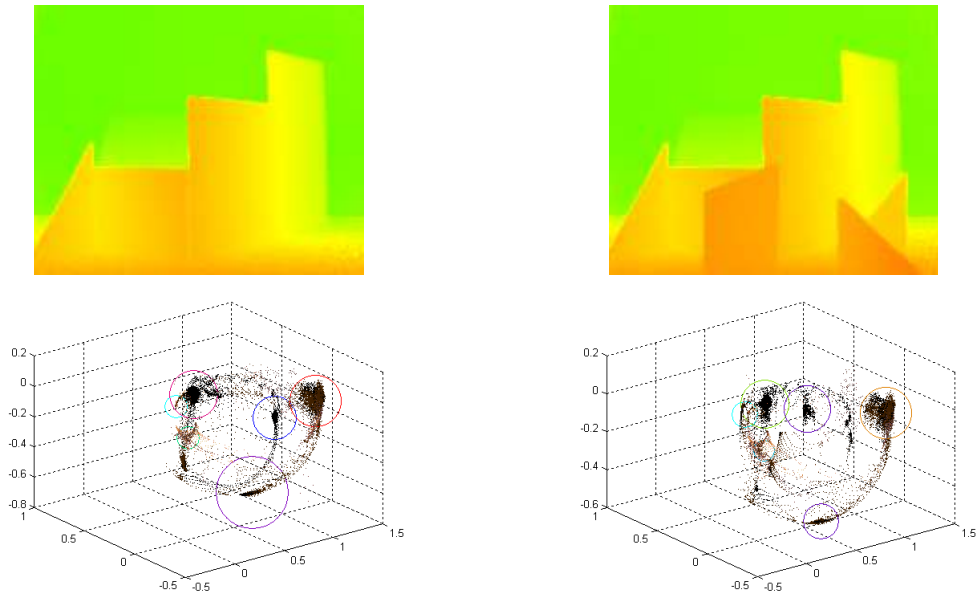


Figure 2.3: Range point clouds as 2D images (top) and the corresponding representations in the  $N \times D$  space (bottom). The range image on the left contains 6 planes and the one on the right 9 planes. In the  $N \times D$  space some points have been grouped to represent points belonging to the same surface.

By computing thresholds for the various clusters detected, we can conclude which points belong to a surface and which do not. The 3D points that do not pass this selection process represent either subsets of very small or very complex surfaces, or elements influenced by error.

Following this approach, all the points included inside a cluster representing a surface are mapped back into the 2D image space, leaving the missing pixels empty. This in itself does not constitute an impediment, as our goal is not to generate complete smoothed images, but to supply to the next surface extraction processing stage a more precise set of information. Thus, the reasoning is that a surface could be better detected from a marginally lower number of correct points than from a larger number of points, strongly influenced by noise.

The main challenge for the denoising as presented so far, is the selection of a correct thresholding function. This function would be best represented by a distance measure that discards points for which the computed value surpassed a certain threshold.

A possible good start is represented by the square error function [25][62] that tries

to minimize the total intra-cluster variance:

$$V = \sum_{i=1}^k \sum_{x_j \in S_i} (x_j - \mu_i)^2 \quad (2.3)$$

Where there are  $k$  clusters  $S_i$ ,  $i = 1, 2, \dots, k$ , and  $\mu_i$  is the centroid or mean point of all the points  $x_j$  in  $S_i$ .

As the square error function is still a selection function that attributes each point to a cluster  $S_k$  based on its measure, it would have to be modified in order to reject values that are too far from any centroid.

Furthermore, the  $k$  means clustering algorithm has the drawback that the gradient descent search might lead to a local minima, and not to the optimal solution. Therefore, we could implement an equivalent formulation of the measure as a trace maximization problem with special constraints as presented in [25]. Under these circumstances, the maximization problem reaches global solutions based on the gradual relaxation of the constraints. Thus, variations of the  $k$  means method could be applied as used in computer vision tasks for image segmentation, with additional modifications supporting error filtering.

Besides the Euclidean space, distance functions for clustering and noise filtering might consider other attributes like color, intensity or even pixel-coordinate weighted distance [51] in order to improve the results of the grouping process.

A particular denoising approach that incorporates many of the previously discussed elements and, in addition, presents a high tolerance to noise is presented in [49]. The approach applies a non-parametric kernel density estimation method with mean-shift clustering on the 3D point cloud.

Given the point cloud, the algorithm computes an unknown density function  $f(x)$ , where an estimation of  $f(x)$  is provided by:

$$f_{est}(x) = \frac{1}{h^3 N} \sum_{i=1}^N \frac{\Phi(x - p_i)}{h} \quad (2.4)$$

Where  $\Phi$  is the kernel (Gaussian) function and  $h$  is the kernel size.

Other similarities to our previous propositions can be found, as the current denoising algorithm takes the computed normal directions into account. Furthermore, each point receives an associated likelihood, representing the probability of it being located on a sampled surface. As such, the filtering method described in [49] eliminates different noise amplitudes by means of thresholding. As suggested previously,

the remaining points would represent the base for a more accurate reconstruction of the underlying surface.

The main difference between this method and our proposed approach consists in the idea of handling the information consistently as a 2D range image instead of a 3D range point cloud, for the filtering, and partially the surface reconstruction. One of the advantages we hope to gain this way is a speed-up of the filtering procedure.

Alternative implementations for denoising based on decomposition of a signal using complex-valued wavelets are presented in [36]. The algorithm maintains the important phase information in the signal given by the image, while at the same time automatically determining a threshold for trimming/shrinking wavelets corresponding to noise.

This approach has the advantage of avoiding the use of distance metrics like the *root mean square* (RMS) measure, which although widely used in image processing tasks, might not be appropriate for every circumstance. Furthermore, many metrics for evaluating the quality of image reconstruction have minimal or no known correlation with the human visual perception, which is almost exclusively the ultimate performance estimation system at our disposal.

However, a feature that appears to be very important in the human perception of images is *phase*. As phase data inside an image contains most valuable information, it suggests the possibility of obtaining benefits, when applying methods that eliminate only the error phase and maintain the perceptually important phase information in the image (2.3).

In order to preserve the phase data and at the same time filter out the noise, the algorithm first extracts the local phase and amplitude information at each pixel based on an implementation of the continuous wavelet transform. Inside the newly generated ND space, noise amplitude distributions are estimated at each scale level. This allows to further automatically deduce the shrinkage thresholds to reduce the magnitudes of the filter response vectors without influencing the main phase.

The mentioned automatic thresholding process [36] is based on the knowledge that wavelets have compact support. Thus, a mapping of the image into the wavelet coefficient space would result in a strong localization of the coefficients supporting important features, whereas the coefficients resulting from noise would have a random distribution over the ND space. In other words, inside this domain the energy from the main signal would be based on a cluster of variables that would be easily detectable, thus allowing a correct noise filtering.

Another important category of filtering techniques that we would like to explore is represented by *anisotropic diffusion* [44]. These methods focus on reducing the noise levels inside an image, without influencing significant parts usually represented by boundaries and other detail elements important for the correct perception of the visualization.

The general algorithm enables a diffusion process that iteratively filters the image space by a locally adapted function. As such, an array of increasingly smoothed images  $u(x, y, t)$  is generated, with  $t \in (0..n)$ , where  $u(x, y, 0)$  represents the original image and  $u(x, y, n)$  the image after the multiple filtering steps.

Most diffusion implementations are developed by using discretized continuous partial differential equation (PDE), which can lead to the development of different kernel filters. This already generates a decision problem, as scientific algorithms for determining the optimal kernel filter are not available. Most diffusion filters use kernels selected based on the personal experience of the research team with the various types of image data. Additionally to the selection of the kernel, another difficult task is the deduction of the edge-stopping criteria, both of which will be further discussed in the following paragraphs.

The anisotropic nature of the diffusion refers to the lack of homogeneity in various local areas of the image. In other words, intensity discontinuities might be present in the images, discontinuities that need to be considered and preserved by the algorithm. Figure 2.4 presents such a discontinuity where the neighborhood of a pixel includes intensities from two different populations. Thus, the diffusion algorithm that needs to compute the adjusted pixel value  $P'$  for  $P$  has to consider the fact that part of the right most neighborhood belongs to another population. Once this fact is recognized, the algorithm should be capable to adjust its function in order to give a lower weight to the pixels representing another intensity group. A failure to detect boundaries and handle them accordingly usually leads to a reduction of the noise levels, but also the undesired effect of blurring the sharp features, fact particularly observable when using an unmodified low-pass filter.

Besides edge removal, shortcomings of the basic algorithm presented in [44] include a weak error reduction performance on large images, and the generation of strikingly divergent solutions from similar initial images.

Therefore, the necessity for creating a diffusion filter tuned to the particularities of our 3D range images becomes clear, as it did in the case of denoising. To create an anisotropic diffusion algorithm capable to handle our dataset and generate the de-

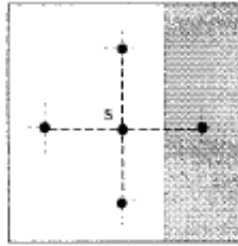


Figure 2.4: Random pixel and its neighborhood near an intensity discontinuity

sired results that would enable for a better surface detection inside the range image, we consider multiple known approaches as a starting point.

An algorithm that considers improving the performance of anisotropic diffusion on large noisy images is presented in [39]. At the same time, it enables the preservation of edges by solving a nonlinear diffusion equation governed by the norm of the image gradient. This edge-strength function is then later supplied to the anisotropic diffusion algorithm, as prior knowledge about the image.

The presented idea also has the advantage of being tested on grayscale images, which can be considered as an equivalent to our normalized distance grid.

Similarly, in [61] an adaptive kernel filter is considered as a natural generalization of the Perona-Malik [44] version, where a parameter  $K$  controls the strength of the filter at pixel level. As a direct result, images processed with this technique receive a strong diffusion effect in areas with same intensity populations, and weaker effects near edges. The parameter  $K$  can be adjusted automatically such that highly-textured regions also receive a powerful diffusion effect, resulting in powerful noise reduction in error prone areas.

The edge-stopping or diffusivity criteria for the anisotropic diffusion is a feature of the algorithm that controls the iterative process, and thus the degree of image smoothing. As such, it has an important impact upon the generated results. Much like the kernel filter functions, the selection of the edge-stopping function in most applications is an ad-hoc procedure based on previous observations.

To counteract this, [5] and [50] present alternatives for automatically computing a close to optimal edge-stopping function by combining fields like Bayesian inference and statistics.

In [5] a correlation between anisotropic diffusion and robust statistics is created.

Robust statistics studies estimation problems in which the data contains gross errors, similar to noise distorted images. In this setting, an original image  $F$  is interpreted and the resulting estimate is viewed as a piecewise smooth image  $G$ . In the ideal case, the generated image  $G$  follows the equation:

$$G + \nu = F \quad (2.5)$$

Where the error  $\nu$  can be considered as a corruption added to the original image by zero-mean Gaussian noise with small variance. Furthermore, the desired output image  $I_o$  would need to satisfy an optimization criterion that minimizes the difference error.

The procedure considers a statistical interpretation of the the PeronaMalik diffusion equation [5], which can be seen as the estimation of a piecewise constant image from a noisy input image. The here proposed edge-stopping function is based on Tukeys biweight robust estimator and provides us with an automatic stopping criterion and a powerful edge preservation method. Additionally, the process allows the detection of edges based on the simple highlighting of the outliers.

But novel edge-stopping functions can also be derived from image statistics by automatically learning function parameters based on previously evaluated training images. Although this approach [50] follows a different path from the previous one, it does supply us with an automatic edge-stopping function, crucial requirement for our on-line preprocessing on range images.

For Equation 2.5, the algorithm tries to determine a generative model for the image pixels  $F_i$ . In order to achieve this, such models are learned from intensity and spacial image statistics. More precisely, a statistical model is deduced from training data presenting pairs of noisy and smooth images, and further utilized to derive a statistically motivated edge-stopping function.

As a related topic, [56] presents a generalization of anisotropic diffusion from image processing to surfaces, by considering the vertex normal vectors as representative to the surface fluctuations. Following the proposition that these normals are the base for a generalization of diffusion techniques in 3D, the subsequent surface deformation can be presented as a process involving the filter application on the normals and a topology preserving deformation of the surface based on the new normal vectors. This leads to the generation of planar patches united by high curvature creases.

While overall focusing on the same topic as us, this approach does not cover our goals of gaining advantages by making use of the particularities of range images and corresponding image processing techniques. Nevertheless, using anisotropic diffusion on surface normals throws us back in the region of denoising methods, suggesting



for our range images a differentiated approach depending on the point attribute: distance, intensity, normal etc.

Furthermore, this idea is explored again in Section 2.3, as surface complexity reduction and elimination of undesired noise features introduced in the mesh model after the multi-mesh merging are topics that would benefit of a surface smoothing technique.

Additional elements that could aid the reduction of the noise present in the 3D range images include histogram equalization and salt-and-pepper noise filtering [51]. Based on the histogram of an image, methods have been developed that better distribute the color values in such way that would generate a more even distribution of the gray levels. This is particularly helpful in the correct detection of edges, as these might be enhanced by creating a stronger visual contrast between neighboring surfaces. Moreover, salt-and-pepper noise reduction relies on the detection and elimination of single dark pixels in bright regions, or single bright pixels in dark regions, a procedure of fairly reduced complexity, but with a positive impact on the point clustering.

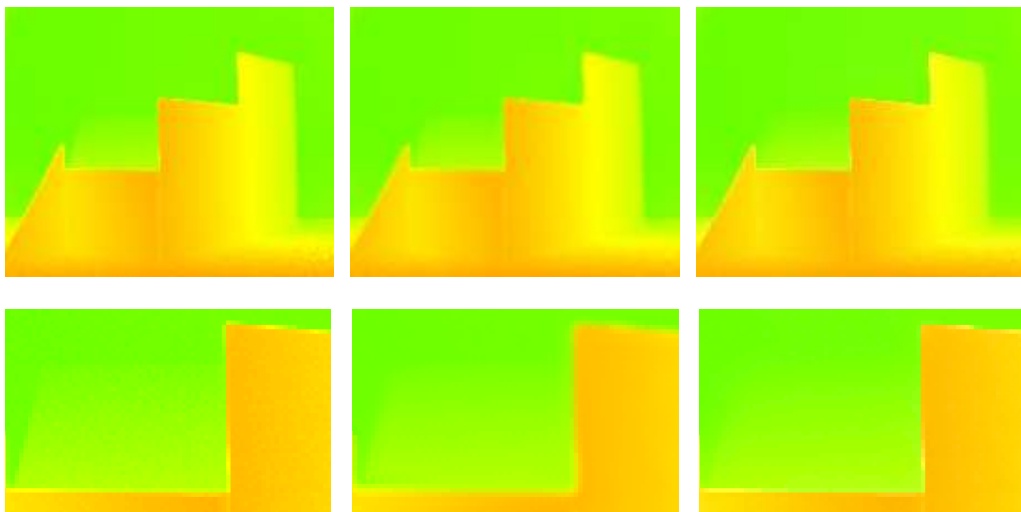


Figure 2.5: Preprocessing filters applied to a range point cloud containing 6 major surfaces: initial 3D range image depicted through a warm color mapping of the distance values (left), range image after anisotropic diffusion (middle), range image after denoising (right). The images on the bottom are close-ups of the corresponding ones on top.

Filtering tests have been already executed on multiple range images. Images 2.5 and 2.6 present some preliminary results. With both filtering methods an increased smoothing of the surfaces is noticeable, and thus an important reduction in the noise levels present in the image. The best results are obtained in areas very close or far

from the sensor, where the initial noise levels are clearly visible on the normalized depth representation.

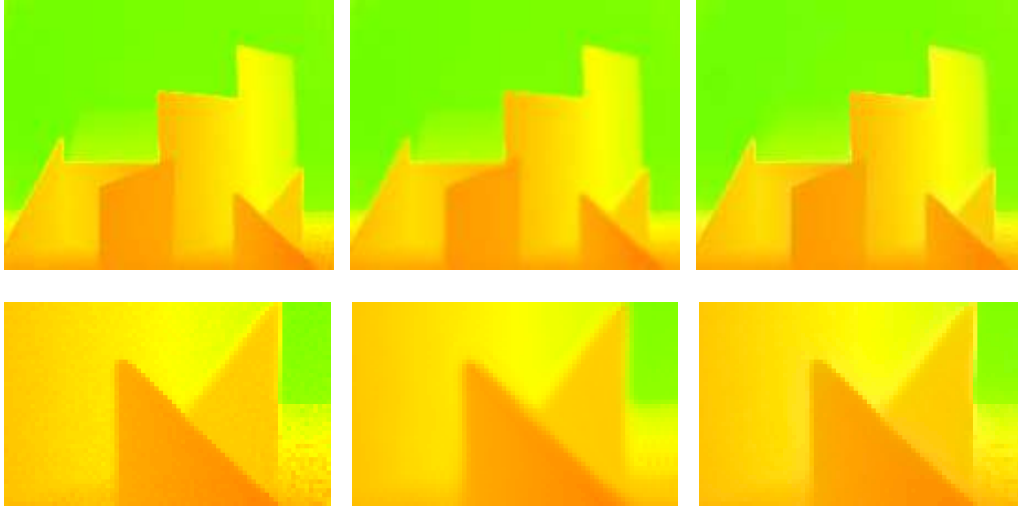


Figure 2.6: Preprocessing filters applied to a range point cloud containing 9 major surfaces: initial 3D range image depicted through a warm color mapping of the distance values (left), range image after anisotropic diffusion (middle), range image after denoising (right). The images on the bottom are close-ups of the corresponding ones on top.

However, the figures present only a partial success, as both methods have undesirable side effects. In the case of the diffusion method, we notice a softening of the surface edges. At the same time, the basic denoising filter manages to reduce the visibility of planes that are very close to the maximal measurable distance. Both these effects may influence the plane/surface extraction process from the later stages and enable misinterpretations of the range image that can result in undetected or incorrectly fused surfaces.

One of the techniques that in our view would ensure a measure for the performance of the filtering algorithms is edge detection (Figure 2.7). Established edge detection methods (i.e. Prewitt, Sobel, and Roberts operators)[10] applied to the 2D distance images can emphasize the addition or disappearance of any surface boundaries, thus suggesting the strength of the filter based on the edge preservation. More precisely, by computing a matrix difference between two images obtained by using an edge detection method on the original range image and the filtered one, and evaluating the resulting grayscale image, one could express the proficiency of a proposed filter.

Methods for avoiding false edge detection due to remaining noise levels, as well as edge enhancement methods, can also be considered [32]. Parts of these methods

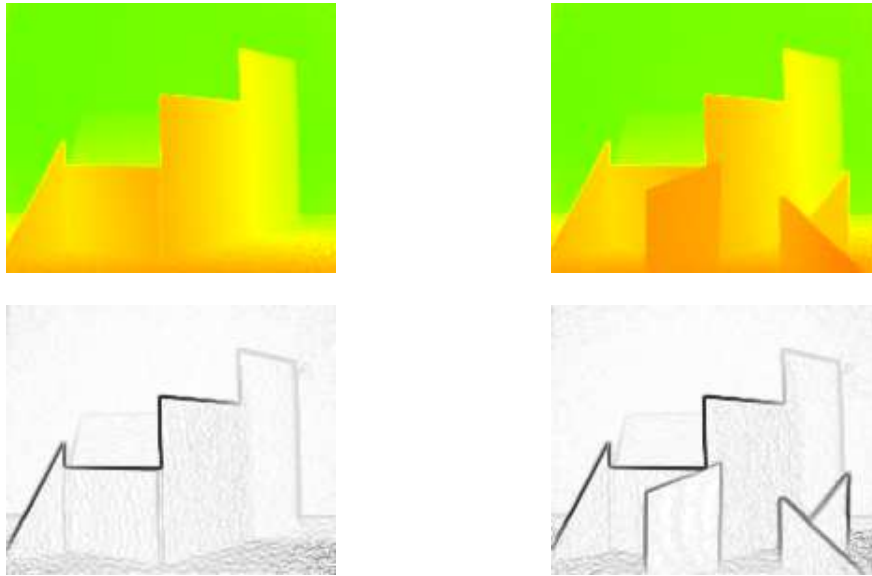


Figure 2.7: Laplacian edge detection results on range point clouds

might be implemented for the surface extraction or image segmentation algorithm (discussed in Section 2.2), but as they are not the topic of this thesis, they might be used as given and not thoroughly explained in these pages.

Other performance measurement metrics for the filtered range images will be represented by applying well established algorithms, like *region growing* [46][58][61], in order to deduce which methods enabled for a better detection and estimation of the 3D surfaces in the datasets.

The preliminary results presented in this subchapter constitute a promising first step for the undergoing research. However, this area requires more implementation efforts and evaluation of the proposed ideas. Additional possible research paths that will be investigated in the area of noise reduction include:

- Various combinations of the filtering methods proposed in this section will be evaluated for performance. The main ideas include: different filter types or filter configurations for each attribute (distance, normal orientation, intensity, shape etc.), sequential application of filters and combinations of these.
- Our research, as well as sources like [24][12], show that the noise levels introduced along the view direction is proportional to the distance to be measured. Inspired by the maximal cut-off distance for these sensors, we want to reduce this threshold by relatively eliminating every point with a value close to the maximal recorded distance. By this we hope to eliminate the most error prone

points, especially as the maximal recorded range is usually the sensor cut-off distance.

- As noise levels in range sensor data are proportional to the scan distance, we propose the distribution of the range points on multiple layers according to distance and local topology. On the generated layers, one can apply local denoising methods specially configured for the particular point sub-cloud. Finally, the denoised data segments are merged, thus forming the resulting range image.

Even with the error reduction and noise filtering presented in this section and applied as a preprocessing step - or even postprocessing on the fused mesh (Section 2.2) in order to minimize merging errors - undesired influences will not be fully eliminated. Under such circumstances, uncertainty visualization methods (Section 2.4) represent a viable technique for correct visual interpretation, in the presence of leftover noise.

## 2.2 Surface Reconstruction Through 3D Mesh Merging

After applying a preprocessing step for the 3D range images in order to reduce the noise levels introduced by the TOF sensors during the scanning stage, we now would like to turn our attention to another important step in SLAM and surface regeneration - the merging of surfaces.

The detected surfaces inside one 3D point cloud represent only a small part of the global puzzle - the environment that should be virtually reconstructed. For example, to obtain a 3D visual representation of an entire wall of a building, chances are that a sensor would require multiple scans to capture the entire surface. These separate scans represent multiple 3D range frames, which additionally have the property that the scan position, direction and orientation might have changed between successive sensor sweeps.

Before we can investigate the algorithms already available for merging surfaces (more exactly mesh representations), we need to establish two elements: which are the surfaces in the range images and what is the correlation between them? There are multiple algorithms that offer us persuasive answers to these questions, most of them being shortly presented further on.

As surface extraction and point cloud/surface matching are not topics in this paper, we will only briefly cover the most important and performant methods, as well as state that our future research will employ a standard implementation for these methods.

To achieve a close to correct extraction of the planar and/or curved surfaces from our filtered range images, we would like to use an implementation of the region growing algorithm [46][58][61] or an image segmentation method [41]. While the region growing algorithm is widely used in the robotics research community, it has the disadvantage of requiring strong seeding points in order to achieve good results. On the other hand, image segmentation techniques have been used and improved for decades, allowing us to choose from a wide variety of proven approaches. Furthermore, as our previously explained filtering ideas should maintain and even emphasize the structural edges inside the range image, the use of an edge-based image segmentation technique would be preferred.

Besides these approaches, other surface generation methods rely on: generating a collection of maximally large NURBS patches that are connected through a Catmull-Clark limit surface [45], combining smooth density functions and ridge extraction methods to reconstruct surfaces based on maximal local density [52], and many more [14][13]. These provide us with alternatives and the possibility of selecting an imple-

mentation that compromises between performance and execution time.

Most of these algorithms have the common characteristic that once the points that make up a surface are deduced, we can apply a tessellation of the range image segments by triangulating over the nearest neighbors. Triangulation over step discontinuities or gaps is achieved by the disconsideration of triangles with edge lengths higher than a previously established threshold.

To additionally improve the performance and accuracy, we might consider applying subsampling in the case of high density point cloud areas, as well as the reconnaissance and elimination of redundant information.

Detecting which points inside a range image represent a surface is only the first step. Additionally we require correspondence information, either between generated frame meshes or between point sub-clouds, to be able to merge the recognized surfaces.

For computing matches between point clouds, the traditional *iterative closest point* (ICP) [3][47][63][2] is one of the mostly used algorithms for virtually reconstructing in-door and urban outdoor environments (Figure 2.8). Modern variations of ICP may consider minimizing error metrics that do not solely rely on Euclidean space, but instead incorporate attributes like normals, colors, edges/shape, a.o. An aid in this process can be represented by the computation and evaluation of the sensor's pose between consecutive scan. Based on the estimated positions and orientations, one could quickly detect 3D point correspondences in a non-iterative way. However, the computation of the translation and rotation of the robot-mounted TOF sensor is typically very unreliable, as position sensors usually lack precision and fail to capture extraordinary events, as for example wheel spin or side trail.

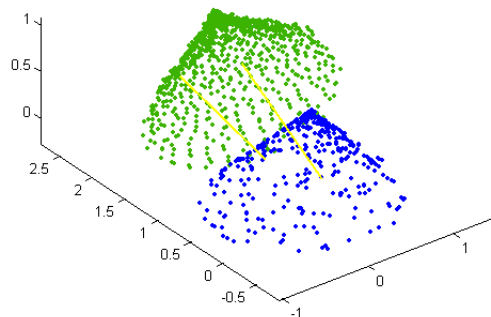


Figure 2.8: Two point clouds and correspondences computed with ICP

Matching surfaces from two different sets, implemented as meshes or planar patches, is another problem thoroughly explored by different research communities. Methods for computing surface correspondence take into consideration a large variety of features like attribute-graphs, shape-factor, area ratio, curvature-histogram, inter-surface relations or combinations of these [19]. A clear advantage, that speaks for a surface/patch based matching, is the reduced computational expense of most of these techniques.

Once surfaces are extracted from range images and the correspondences for these segments are computed, we can divert our attention towards the mesh merging process. The merging approach we would like to implement is similar to the one presented in [14], where after processing one range image at a time, the results are combined with the already processed information in a simple additive way. More precisely, in our research we hope to achieve a high-accuracy iterative fusion of the frame surface meshes. Each newly analyzed range cloud will add none, one or multiple surface meshes to the globally accumulated mesh representation. The surface meshes can be directly generated from the corresponding 3D points of the range image, but in order to simplify the mesh and eliminate redundant information, the selected point set will be down-sampled. Still, this iterative process leads to a multi-mesh representation that gains precision, but also complexity, with every added partial surface.

The problem of linearly increasing complexity of the overall mesh will influence the rendering performance of the reconstructed environment. To reduce these effects, multiresolution methods will be discussed in Section 2.3 as a measure to automatically adapt the complexity of the visualized environment to the size of it, and at the same time consider the computation reduction for areas that are out of the view frustum.

As the generation of multiple resolutions, together with tasks like hole detection and filling, usually present themselves as computationally expensive, these refining procedures might be executed only on demand, based either on particular thresholds or simply by limiting their execution to every  $n$ -th merging iteration. Moreover, we consider the introduction of a final optimization step that would shift the execution of certain delayable and low priority processing events towards the end of the pipeline; in case of time pressure under online circumstances, these might be omitted or simplified. A good example for such delayable events is represented by hole filling tasks, which in many cases can be performed once, after the global surface is generated by considering all available 3D range point clouds.

While we have discussed the overall implications of the mesh merging procedure we would like to apply, our main interest is focused on developing an actual core

merging algorithm. Supplying an additional set of meshes for each point cloud image and merging each of these meshes with the corresponding global one are not atomic tasks, but the core merging algorithm would be solely represented by a strategy for merging two corresponding 3D meshes.

A good starting point towards understanding this procedure, its complexity and possible complications that might arise, is represented by [57]. The article covers an approach for combining a set of range images obtained from multiple TOF sensor scans of closed surface objects. The acquired collection of meshes is merged into a single polygonal mesh during an incremental process, where at each step a new range image generated mesh is added. The results of the fusion is a complex mesh that entirely describes the topology and surface of the analyzed objects.

The idea exposed in [57] is strikingly similar to the one considered at the beginning. Initially, preprocessing steps ensure that the mesh representation considers a correct topology. Once the final meshes are established, the distance between the two triangle meshes that need to be fused is minimized. A particularity of the method that might already represent a feature that we need to overcome in our research, is that merging is executed for two overlapping meshes only, which is not always the case for consecutive range images of an environment.

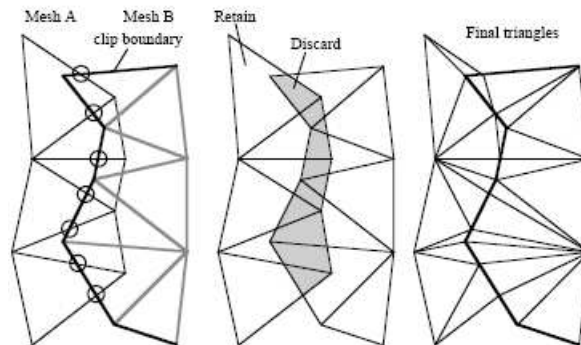


Figure 2.9: Merging of two partially overlapping meshes: (left) initial meshes and the established clip boundary, with the edge intersections highlighted, (middle) overlapping triangles that will be discarded from the first mesh, and (right) final merged mesh with additional vertices for previous edge intersections

Once two meshes are prepared, their correspondence and a certain level of overlap are established, the meshes are fused by the means of an intuitive process called *zipping*. This process is presented in Figure 2.9 and consists of three main steps:

1. Remove overlapping portions of the meshes - redundant triangles are removed on both meshes until there are no more triangles in any of the two meshes that



completely cover a part in the other surface.

2. Zipper the two meshes - considers edge intersections between the triangle boundaries of the two meshes and inserts additional vertices at those positions.
3. Remove the small triangles introduced - the merging process may create small triangles; to avoid this, triangles with an area or edge below a threshold undergo a process where a vertex is removed and all the surrounding triangles of that node are joint, thus producing a larger triangle.

A question that might appear already at the first step, is how to determine the existence of an overlap of two 2D manifolds in 3D space. The solution for this is based on the computation of the normals for the surfaces, which are used to detect an intersection between the normal direction at a point and the second mesh representation. This enables to detect if an overlap exists even when the surfaces are slightly gaped, for example due to scanning errors.

Figure 2.10 presents two meshes that are shifted along the normal direction. To overcome this in the detection of an overlap, a 2D manifold is introduced, based on the normals of the mesh boundary. This third mesh is used to extend the margins of the surface and detect any superposition between the corresponding meshes.

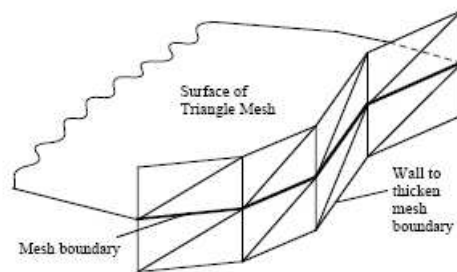


Figure 2.10: 2D manifold representing the extended boundary of the 3D surface mesh

However, the fact that the creation of the global mesh structure is relatively slow and not real-time and that we are presented with closed surfaces, leaves room for research and adaptations that might make this technique applicable for online mesh generation of in-door and outdoor environments.

Contrary to the previous article, [53] concentrates on merging non-redundant surfaces that do not present an overlap. In order to do so, the Venn diagram is computed for the range images by finding the contents of the canonical subset that is sustained by the correspondences in the mesh sets. Furthermore, the orientation of the triangles

is also computed, allowing for a correct joining of the resulting meshes.

Figure 2.11 presents the merging procedure of two non-redundant surfaces. The empty area between the meshes is interpolated by a constrained Delaunay triangulation. While the authors focus part of their efforts on removing the redundancy, our research will focus on combining the methods of [57] and [53] in order to detect overlap and apply the corresponding algorithm, without trimming or extending the initial meshes.

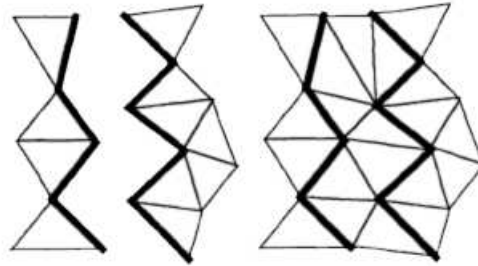


Figure 2.11: Boundary interpolation between two non-overlapping meshes in 3D space: (left) initial meshes with nearby boundary highlighted, (right) merged mesh after interpolation

Additionally, as a final optimization step, the generated joint mesh is analyzed and different *levels of resolution or detail* (LOD) of the surface are generated, based on various areas of interest. This is of a particular importance for our research of multiresolution methods, presented in the next section.

Of further interest for us, [13] presents a complete and detailed solution for extracting range images and reconstructing complete virtual representations from them. Once the triangle mesh is generated, the signed distance contribution is calculated by means of ray tracing from the viewpoint (initially, TOF sensor) through each voxel near the range surface. These rays might eventually intersect the global triangle mesh, and allow for weights to be computed. Based on these measures, in chapter 6.1 the merging process is thoroughly explained, the end result being the extraction of a zero-crossing isosurface from the volumetric grid.

Understanding and following alternatives for all the steps of surface reconstruction is of high relevance for a successful research. Nevertheless, much of our attention goes towards chapter 7.1 of [13], dedicated to optimizing the overall process and making it computationally inexpensive, as this is of vital to achieving a real-time execution.

Previous research, as shown in [13], has presented that the parsing manner of the

range image and the 3D space is a critical aspect, which influences efficiency of execution. As this process will be required in multiple sections, focusing on its improvement is a clear priority. To avoid random access on either side, the data storage has to be optimized for the given task, such that a direct mapping between the range image and the 3D data exists.

While there are still task-driven particularities to the merging of TOF sensor generated point sub-clouds in the form of meshes, these challenges and possible improvements will represent the area of main interest in implementing a functional mesh fusion technique.

A particular problem is represented by the presence of holes in meshes, or more precisely the detection and correct elimination of them. As usually there's no way to establish which areas represent a gap/hole, and which are simply scanning errors based on the point clouds or the point-based meshes. A simplistic solutions for overcoming this is given by establishing a threshold for the inter-triangle area, which governs the minimal margin for empty regions (Figure 2.12).

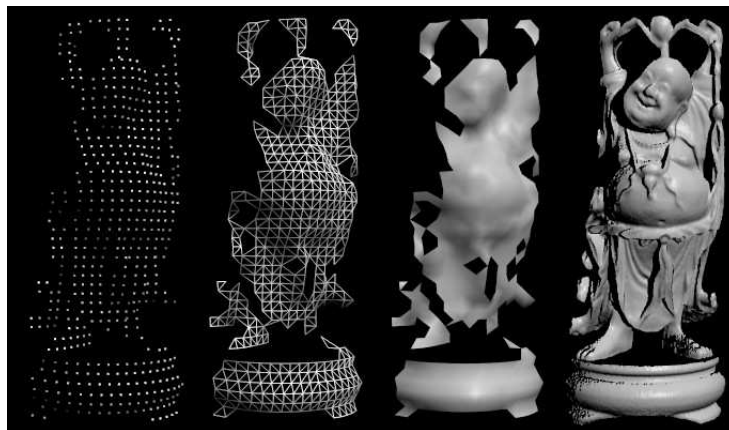


Figure 2.12: Processing steps from point cloud to surface mesh: (left) sub-sampled 3D range point cloud (middle left) triangulation of the closest points (middle right) mesh surface rendering with faces computed only for triangulations with an area under a given threshold (right) final high resolution surface rendering after eliminating mesh holes

Methods concerned with partial reconstruction of holes in meshes generated from point cloud sets of various buildings and architectures are presented in [53]. But mesh-based detection and elimination of surface holes usually operates on the reconstructed mesh, by tessellating over the boundaries between regions considered to be empty. While this approach might be sufficient for most planar surfaces, it performs very poorly in the case of highly non-planar gaps.

As an alternative, the idea of volume-based hole filling is presented in [14], where all the points of the mesh are classified into three states (unseen, empty and near surface) in order to better detect possible gaps.

Still, both mesh and volume based algorithms have difficulty detecting the difference between a real hole in the data and one generated by undersampling. This means that such an algorithm would either detect both gaps and errors as holes, and thus not correct the errors, or the other way around.

A volume oriented approach is only one possibility, nevertheless. Paper [8] presents multiple hole filling techniques besides interactive or post-processing the generated mesh, methods that involve surface evolution over time or the use of radial basis functions (RBF) to compute a weighted sum of the points, and obtain the surface based on the level set of the last RBF. Additionally, [8] focuses on *inpainting*, a term that suggests the modification of images such that these changes are undetectable. In our specific case, this would apply as filling in the missing regions with replacement information that allows for further processing.

All these methods for detecting holes and their nature, while also handling them appropriately, constitute a topic we want to further investigate for our mesh merging.

The implementation of the mesh merging algorithm and the virtual reconstruction of the overall TOF sensor scans does not by far represent the final checkpoint in our process. In the next sections we will emphasize the methods we would like to use to tackle the increasing complexity of the surface representation, as well as solutions for supporting correct interpretation and analysis via integrating uncertainty visualization into the surface rendering.

## 2.3 Mesh Simplification and Multiresolution Representation

Even with the elimination of redundant points in the previous step and the implementations of information architectures that enforces quick data access and retrieval, the complexity of the mesh representation will still increase linearly with the number of added vertices. This can in turn lead to a highly complex global mesh that needs to be visualized interactively, a task that becomes increasingly non-trivial with the merging of additional information.

To counteract this, we concentrate on devising a multiresolution representation which is particularly tuned to sustain the requirements of our endeavor. In other words, our goal at this stage is to conceive a simplification and subdivision scheme for our final mesh, such that it incorporates a specific low-pass filter, and at the same time preserve boundary shape and area for the global mesh set.

A first step we consider is the simplification [21] - as much as possible with the side-effect of smoothing - of the surface rendering. By these means we hope to obtain a good approximation of the original model, generated by reduction of the number of vertices and polygons. Methods that can achieve this are usually based on appearance attributes [11], like position, curvature and color.

As our original information has basically only the position data incorporated in the point clouds, we consider this attribute to be most relevant. Starting from the generated global mesh, which is basically a set of triangulated point clouds (Figure 2.12), algorithms have been devised [11] to filter part of the supporting vertex set in order to obtain surface approximations based on local simplification operations.

The local simplification refers to edge-collapse methods that reduce the number of triangle faces, edges and vertices by eliminating, through deletion or merging, vertex points from the surface. A particular importance should be given to the order in which the simplification operations are executed. Most methods apply an error metric, be it local or global, that mirrors which removal operation can be executed next with a minimal information loss. Thus, the vertex deletion is performed in order of increasing error, assuring that relevant features receive a low removal priority.

Furthermore, mesh simplification can be categorized into strong and weak, depending whether the affected vertices are simply a subset of the original high resolution mesh, or if they are allowed to be positioned freely.

Removing nodes and information is not the only way to simplify, or smoothen, a

surface representation. An alternative is presented in [56], where after a generalization of anisotropic diffusion for 3D space, based on vertex normals, mesh vertices are repositioned and the surface is deformed to better support a smooth transition between normal values. At the same time, creases are detected as discontinuities in the normal space.

At the root of this approach lies the supposition that normals are a natural generalization of points from 2D domain, as they manage to describe the manifold more completely by incorporating additional information. Furthermore, normals are coupled to the surface, and thus they manage to guide the mesh in correspondence with their values.

As such, level set surface models can be devised that allow the complex manipulation of arbitrary topology. A desired side effect, as presented in [56], is that the approach is naturally suited for surfaces, which are generated from measured data, fact that makes this approach even more relevant to our work.

This approach is important for us, as any method that simplifies a surface topology by smoothing it could be used as the base for the error metric of a simplification algorithm.

Approximation takes on an entirely new dimension in [26], where smooth 3D representations are derived by means of planar approximations. Based on the 3D point clouds and applying a region growing algorithm [61][46][58], planar patches are constructed from a maximum underlying set of points that can be approximated by a flat 2D manifold. Afterward, neighboring patches are merged if and only if they rely on the same plane. At the end of this process, a set of planar surfaces is obtained for which the corresponding mesh edges are known. Additionally, these polygons simplify an area of the mesh by eliminating vertices that retain redundant information when on a planar segment.

By using the mapping between patches and mesh areas, certain sections of the 3D mesh can be modified or replaced in a way that strongly reduces the overall complexity. For complex, non-planar regions of the mesh, no corresponding planar surface will be detected and the triangulation in that area is left unchanged. Differentiating between flat surfaces affected by noise and actual sharp features (i.e. corners) is a characteristic that makes this method stand out, as detecting real topology changes and allowing for higher LODs in those areas is of relevance to this thesis.

Figure 2.13 presents the results of the technique from [26], when applied on a 3D range scan of an in-door hallway.



Figure 2.13: Mesh model from 3D point cloud of the Wean Hall at the Carnegie Mellon University (left) and equivalent smoother, low complexity mesh model obtained by planar approximation (right)

We speculate that this approach, coupled with e.g. adaptive subdivision methods [4] would enable us to devise a mesh simplification that supports interactive visualization even for a very complex initial mesh.

Other methods of similar performance in simplifying surface reconstructions are presented in [22], where computer graphics algorithms are employed in order to achieve a powerful structural reduction of the 3D models.

While the simplification of a 3D mesh is of remarkable importance for the real-time visualization of such a representation, it only solves half of the problem. Of course, it is relevant to reduce the number of mesh triangles in order to decrease the computational complexity. However, to obtain a dynamic, view-dependent management of LODs, one needs to be able to retrace the steps and refine an area to an - in our case, initial - high resolution state. To obtain this functionality, the concept of *progressive meshes* (PM) is considered.

Progressive meshes [27][28][37] represent a category of algorithms that deal with the simplification of a mesh structure by introducing a bijective mapping between the original surface representation and a base mesh consisting of a smaller number of faces (Figure 2.14). Usually the base domain is described as the least complex mesh structure, held up by the least amount of mesh vertices.

A concrete approach for a fast and efficient hierarchical multiresolution method is described in [37]. Due to the large size and the complex structure of many 3D meshes, a hierarchical subdivision connectivity remeshing is required to be able to access various LODs, and thus increase rendering performance.

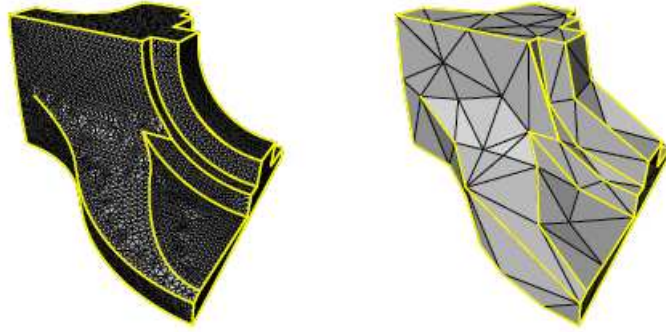


Figure 2.14: Fine (left, 12.946 triangles) and coarse resolution (right, 168 triangles) mesh representation with preservation of edges, darts and corner vertices on the base mesh

The adaptive multiresolution approach considered in [37] employs such a subdivision connectivity, allowing for the coarsening and remeshing of the isosurface, depending on various requirements.

To create a mapping between the original mesh and the base domain, a coarse-to-fine approach is used that follows these guidelines:

- *Coarsening*: simplify the original mesh until a base mesh is obtained, which is defined as the minimal mesh representation that does not incorporate topological changes.
- *Mapping*: a correspondence must be created between every vertex of the original mesh and a face of the base model. The mapping is usually computed concurrently with the generation of the hierarchical structure that supports the multiresolution implementation.
- *Remeshing*: reinsert vertices one-by-one to iteratively increase the LOD, thus enabling the generation of an adaptive remesh with subdivision connectivity.
- *Optimizing*: execute local optimizations for the inserted points. One of the effects that can be obtained this way is a smoothing of the overall surface.

More precisely, let's consider the original surface as  $(P, K)$ , where  $P$  represents  $N$  3D point positions  $P_i = (x_i, y_i, z_i) \in R^3$  with  $1 \leq i \leq N$ , and  $K$  contains the detail information of every level. In order to achieve a mesh hierarchy, the original representation  $(P, K) = (P^L, K^L)$  is successively simplified until the base domain  $(P^0, K^0)$  is reached. All the intermediate meshes  $(P^l, K^l)$  with  $0 \leq l \leq L$  are



homeomorphic and can be obtained by computing:

$$(P^l, K^l) = (P^0, K^0) + \sum_{i=1}^l K^i \quad (2.6)$$

Where  $K^l \rightarrow K^{l-1}$  represents a simplification step. These intermediate checkpoints are extracted by applying an operation called *edge-collapse*, which is basically composed of two elements: the removal of the vertex with the lowest detail cost and the retriangulation of the resulting gap.

A known side effect to vertex removal is the flattening of the generated holes, which in our case, even if desirable as a mean to smoothen the surface and eliminate noise, introduces the problem of area reduction and changing boundaries.

Furthermore, during the vertex reinsertion stage the positions of the 3D points can be affected in order to support particular effects, like smoothing. In [37] this is achieved by computing a distance measure  $E$  between each point of the original mesh and the base triangles. By computing a threshold  $\epsilon$ , where  $\epsilon = E/B$  and  $B$  is the largest side of the bounding box, we can subdivide the coarse representation by applying a variation of the Loop method.

While the Loop subdivision has the advantage of low computational expense, other subdivision schemes like those devised by Doo-Sabin [17][16] and Catmull-Clark [9] must also be considered for this thesis.

Contrary to the previous article, [35] focuses on generalizing multiresolution approaches for arbitrary triangle meshes, but without the use of subdivision connectivity. Not only that the hierarchical structure presented in [37] is replaced by a set of intermediate meshes generated by surface decimation, but combined with the constrained minimization of discrete energy functions, a fast multiresolution approach is devised. Additionally, the method allows for isolation and reduction of noise in a particular band, thus smoothing the mesh.

Next, decomposition and reconstruction operations are defined, that allow for a natural separation between the surface details and the low frequency surface shape. A particularity in this process is represented by the reinsertion method, which considers storing the detail coefficients as difference vectors between the original point  $P_i$  and the optimized one  $P'_i$ . Additionally, the vital topic of constrained mesh optimization is presented and solved in a real-time fashion.

Other methods that introduce variations or extend the limits of multiresolution representation are presented in [48][45][20], all of them having as a common element

the need for controlled topology simplification to achieve real-time human interaction with the model. A slightly different approach for creating a mesh is presented in [54], where 3D point clouds form the source for the implementation of a surface visualization based on an octree representation.

A particularity of our future research related to devising a multiresolution implementation for the reconstructed surfaces is represented by the way we would handle multiresolution dependent mesh boundaries.

As the global surface representation is composed of a set of meshes that might or might not be zippered (i.e. one wall could be stored as a single mesh, unconnected to any others), we have to find subdivision algorithms for mesh edges, or more precisely, for planar surface edges. To achieve this, we distinguish two cases: crease preservation and edge preservation - or area and shape preservation - for complex meshes.

Crease preservation refers to maintaining sharp features in the mesh topology, features that are of high relevance for the overall visualization process. An example of such a crease would be represented by the connected triangle edges that represent the wall corner inside a 3D mesh generated from the scans of two orthogonal surfaces. Moreover, we hope to reduce the informational and visual complexity of quasi-planar surfaces, and at the same time enable the representation of major topology changes with minimal effort. Related to this, [35] presents a simplification approach for shared borders edges that considers a symmetrical subdivision method.

However, edge preservation in a multiresolution scenario refers to a more difficult task of maintaining the outer shape of the mesh, and implicitly, its area. While focusing on this problem, we would like to explore variations on adaptive subdivision schemes [4], which might ensure a constant boundary shape for the generated global mesh set.

Even if all the sections described this far consider the presence of errors in the input data and try to compensate, a perfect noise reduction is highly unlikely due to its stochastic nature and multi-level introduction (Figure 2.15). Knowing where in the surface reconstruction we still have noise and at what intensity, would assist the analysis process of the end-user, be it human or artificial. To facilitate this, the next section focuses on adding uncertainty information to the visualization of the reconstructed 2D manifolds.

## 2.4 Surface Uncertainty Visualization

While in Chapter 2.1 we focused on approaches we would implement in order to reduce the amount of inherent errors that appear in sensor-acquired data, in this section we present methods for visualizing the uncertainty of the reconstructed surfaces.

Uncertainty is presents in all aspects of life, and this is also reflected in the digital world. If we visualize any information that is affected by errors of different nature without including a visualization of the uncertainty present in the data, analysis of these visualizations run the risk of reaching incomplete or inaccurate conclusions. This is the main reason for which uncertainty visualization tries to display the additional error information incorporated in the representation.

All types of sensors created until this point in history are affected to a lower of higher degree by uncertainty, which is uphold by external errors introduced during the measuring stage or internal errors based on the imprecisions of the device itself. This type of uncertainty is called acquisition uncertainty [42] and it usually represents the first point of a digitization process, where error is introduced in the production pipeline (Figure 2.15). These errors can vary in consistency and structure from one scanner to another, a few error types being represented by: precision errors, minimum or maximum truncation values, external noise influence, missing data, etc.

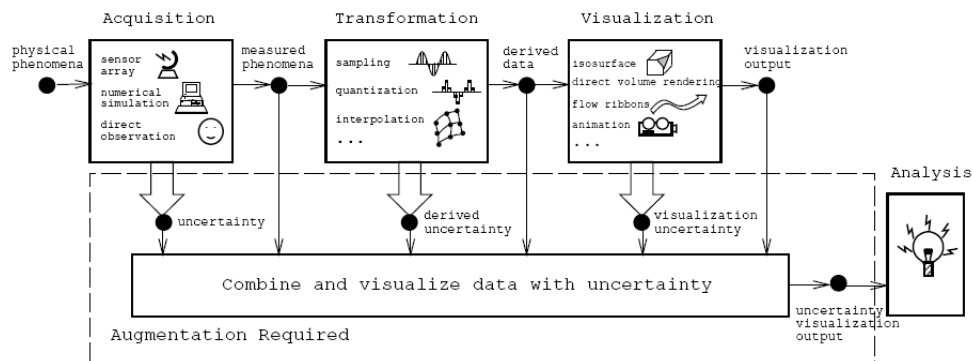


Figure 2.15: Pipeline showing the stages that can introduce uncertainty: acquisition, transformation and visualization uncertainty

The applied sensor technology is of crucial importance to the distribution and variation of the uncertainty over the entire surface. From [14] we know that uncertainty variations in sensor data can be overcome by applying weights that capture the nature of the variation. Such weighting factors include the dot product between each vertex normal and the viewing direction for triangulation scanner, or simply a distance value for TOF sensors, where we know that the error margin for each scan increases with

the distance.

For the case of TOF scanners, the primary source of error (assuming an environment without external noise that acts on the sensor) is given by the travel time of the light pulse, from the emitter to the reflection point and back to the sensor. As our preliminary analysis of the two point cloud datasets (Chapter 2) showed, our main concern for the error levels in the input data will be given by the depth errors along the scanning direction of the beams [13].

Another type of uncertainty can be introduced in the process due to successive transformations of the scanned data: approximation, fusion and derivation of new information from the initial data. This step, together with the data acquisition stage, will probably introduce the highest amount of uncertainty in our surface reconstruction process, even with the error reduction algorithms applied to the initial TOF sensor datasets.

The last stop in the pipeline of uncertainty is represented by the visualization uncertainty itself, an element often underestimated or considered to have limited relevance. This visualization insecurity covers uncertain rendering and illumination of objects. For example, a complex surface with a poor illumination scheme applied to it can have as effect that the viewer misinterprets the topology of the structure or the shape in some areas.

To avoid this, our implementation will support a proper illumination and interactive viewing of the virtual environment. These features might be available also in online processing mode, but if the computational expense surpasses the limit of real-time processing, these will be automatically deactivated in order to analyze the data in an online manner.

Figure 2.16 presents the uncertainty types we expect to encounter in our mesh generation. While  $UL1$  is representing the uncertainty added by the input dataset acquisition,  $UL2-UL4$  are the equivalent of the transformation errors resulted from the algorithms that will be developed as presented in Sections 2.2 and 2.3, and introduced by the second stage of the uncertainty pipeline. Knowing the types of uncertainties we can expect, as well as the stages of our reconstruction process at which they will appear, our goal is extended to not only visualizing the cumulated surface errors, but even to supporting an *interactive selective visualization* of the uncertainties. This offers the advantage of visualizing the effects of one or multiple particular error factors when applied to the reconstructed mesh (e.g. visualization of the surface with the transformation uncertainties only -  $UL2$ ,  $UL3$ ,  $UL4$ ).

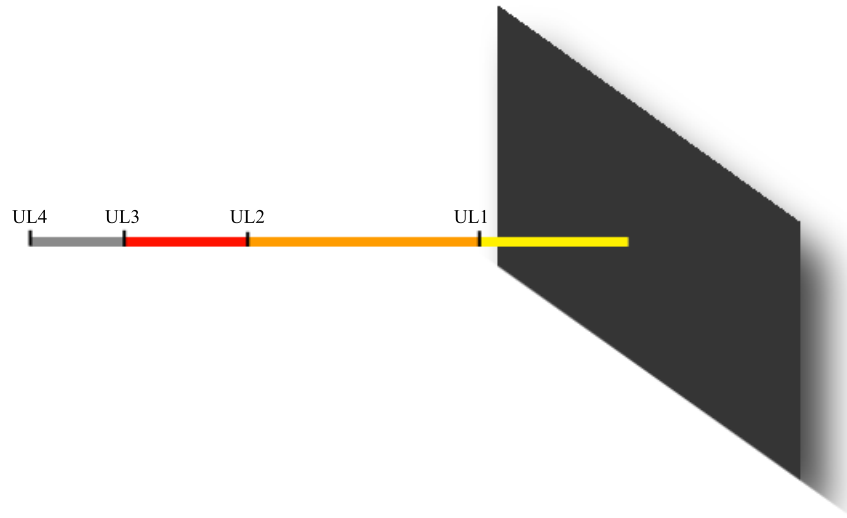


Figure 2.16: The 4 levels of uncertainty in our mesh visualization. The figure presents a gray surface on the right, which can be interpreted as a single planar patch in the entire mesh. On the normal direction to the patch, we represent the different levels of uncertainty in the order of appearance: UL1 - uncertainty added by the noise in the TOF sensor data, UL2 - uncertainty in the computed local patches, UL3 - uncertainty through repeated point cloud matching and merging, UL4 - uncertainty introduced by mesh multiresolution methods

As visualization of uncertainty for surfaces has received an important amount of attention from the scientific community, at this point we would like to present a few visualization methods, as well as new ideas and modifications of previously proven algorithms to the subject of surface uncertainty visualization.

One of the approach categories we would like to avoid due to the lack of correlation between the obtained surfaces and the uncertainties, is represented by techniques that visualize the surfaces and the uncertainty separately. In this case, an extra variable representing the uncertainty is incorporated in the dataset and displayed on a separate subfigure, using any standard visualization method.

In contrast to this idea is the one of *verity uncertainty visualization* [59][55], where new and/or adaptations of existing visualization methods incorporate the generated information and the auxiliary uncertainty data in the same representation. The advantage of this approach is that the facts can be transmitted consistently under this unified representation, but one has to give extra attention to not visually overloading the 2D display area. A mismanagement of the visualization can lead to 3D occlusions, weak distinctive features and wrong conclusions, thus losing the advantages

previously gained.

For verity uncertainty visualization, there are multiple techniques and classifications. The one presented in [42] clearly underlines the separate categories by highlighting the set of variables that the representation may influence: adding geometry, modifying geometry (space as variable), modifying geometry attributes (color, texture, etc.), animation (time-space variation) and sonification (sound variable). The following paragraphs convey existing methods and possible new ideas for visualizing surface uncertainty as mentioned above. Methods for animation and sonification are intentionally omitted, as these might not be desired (i.e. sound) or do not seem to bring significant advantages to our topic.

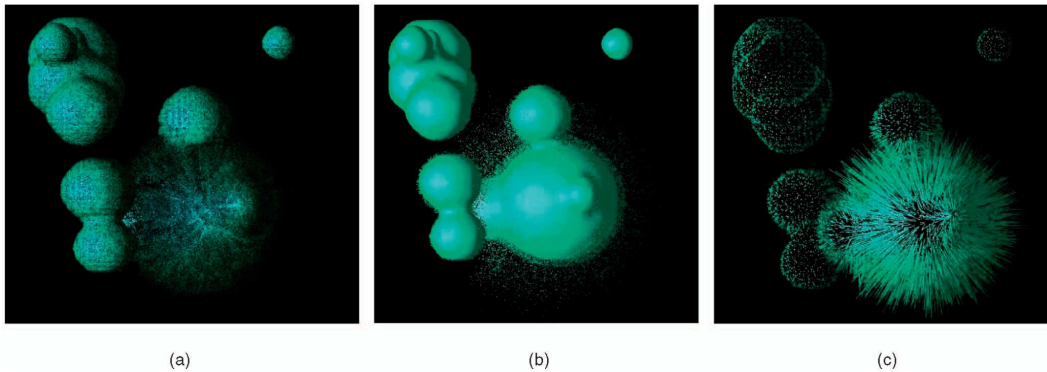


Figure 2.17: Point-based surface uncertainty visualization - (a) opacity modulation (b) Gaussian probability distribution (c) representation of normals with length proportional to the error and without displaying the underlying surface

Adding geometrical elements to the scene and/or modifying the surface geometry is one of the key possibilities for visualizing the uncertainty of the reconstruction. A particularly simple, but effective method is obtained by using points for uncertainty visualization of surface representations [23]. In this case a collection of points can be positioned along the surface normals at a computed distance from the surface, proportional to the uncertainty in that area (Figure 2.17). The normals that would be taken into consideration can pass through various surface points, which in our scenario would most probably be represented by mesh vertices or the barycenter of the mesh faces. Not only that these origin points  $V$  for the normals allow for a sufficient representation of the uncertainty, but they also have the advantage that for the mesh vertices we should hold the possibility of computing the differentiated uncertainty, as presented in Figure 2.16.

The basic algorithm behind the point-based uncertainty representation can be modified and adapted for our obtained surfaces, thus leading to the following pseudo-code:

1. Obtain uncertainty level for mesh vertices  $V$
2. For each mesh vertex  $V$ :
3. Compute the normal  $N$  to the surface
4. Generate two points  $P_i$  and  $P_j$ , situated on the normal  $N$ , with  $P_i$  on one side of the surface, and  $P_j$  on the other
5. Calculate the displacement of  $P_i$  and  $P_j$  compared to  $V$
6. Displace  $P_i$  and  $P_j$  in the direction of the normal at  $V$  and render them
7. End the for loop

There are multiple possibilities for computing the displacement of the points  $P_{i,j}$ . The equation proposed in [23]:

$$Displacement = Rand() \cdot (Uncertainty_{atV})^a \cdot Scale \quad (2.7)$$

Takes into consideration the uncertainty level at the current position (influenced by the falloff  $a$ ) and weightens it by the scale and a random number in order to avoid creating a dense point set that resembles a maximal or minimal uncertainty surface, which would occlude the reconstructed mesh. Nevertheless, in cases with a relatively small number of points this risk is reduced and the randomization term can be omitted.

The method has a complexity of  $O(n)$ , which can be supported in a real-time implementation, even if the number of added points  $P_{i,j}$  is not 2 (one  $P_i$  and  $P_j$  for each vertex point  $V$ ), but 8 (corresponding to both sides of the surface and all 4 possible uncertainty levels).

As an alternative to displaced points, line segments on the normals could be depicted (Figure 2.17-c). The length of each segment would be established by the surface point  $V$  and the displaced point on the normal direction  $P_{i,j}$ . This idea is also suggested by Figure 2.16, where multiple contiguous segments are used to depict the different uncertainties at the surface point. In order to better differentiate the lines from the generated surfaces, as well as the multiple uncertainties present at that surface point between one another, the lines can be rendered using color coding, hue variation or added transparency proportional to the segments' length.

Another geometry based uncertainty visualization method that we would like to explore considers the suggestion of the uncertainty levels via translation or rotation of the reconstructed mesh faces (Figure 2.18). While the translation of the patches along the normal direction is trivial and embeds some of the features already presented for the point-based algorithm, in the rotation of the faces multiple issues have

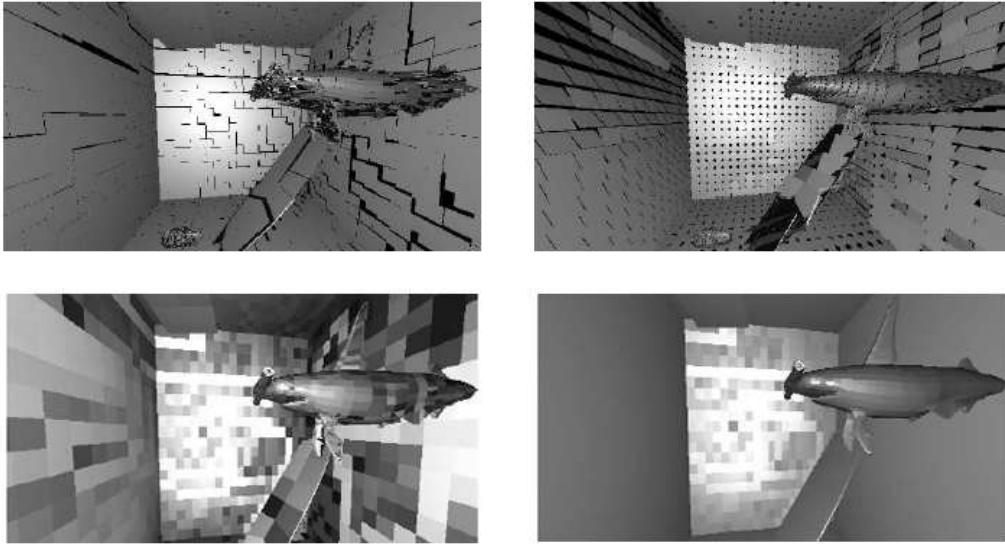


Figure 2.18: Surface uncertainty visualization through - (a) translation of surface patches (b) rotation of surface patches (c) change in the diffusion coefficients (d) change in the specular coefficients - by a value equal to the difference between the real surface and the obtained one

to be considered: axis of rotation - which is usually a fixed direction in space, like one of the coordinate system axes - and overall result of the rotation (while the rotation of the mesh faces might help to suggest uncertainty in quadric meshes, for triangular meshes the results are poor and this method should probably be avoided).

A set of geometrical methods for uncertainty visualization that can be specifically applied on surfaces is suggested in [55]. These algorithms are grouped in uncertainty glyphs, fat surfaces, perturbations and oscillations.

Uncertainty glyphs are symbols or icons that encode error information based on shape and/or geometrical attributes [33](like color, saturation, etc.). A representation where deformed cones and coloring are used as glyphs in order to highlight additional features of a surface is given in Figure 2.19. This representation can be easily adapted for surface uncertainty visualization, with the additional constraint of simplifying the shape of the glyphs in order to support real-time rendering and interaction. As previously mentioned for the line-based approach, different error elements can be visually enhanced by color mapping or transparency.

While shape glyphs add geometry to the reconstructed surfaces, fat surfaces try to apply a technique that covers the entire uncertainty space by incorporating all the possible values into a volume. This visualization can be obtained by using all the left-most and right-most uncertainty points  $P_{i,j}$  in order to render two bounding



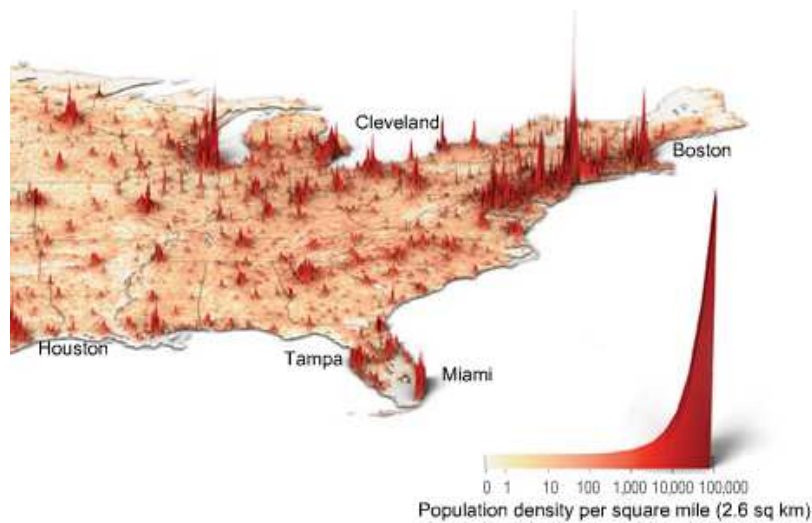


Figure 2.19: Glyph representation of population density per square mile. Geometrical glyphs attached to surfaces can alternatively represent uncertainty

meshes for the uncertainty volume. Such a pair of two meshes can be later connected into forming a closed surface by applying triangular or quatric patches between the corresponding  $P_i$  and  $P_j$  points on the mesh boundaries.

A similar approach that can bring some advantages to uncertainty visualization is the representation of the reconstructed surface, as well as the two boundary maximum error meshes. In order to facilitate the distinction of the different surfaces by the user, the two boundary meshes should be semi-transparent and each surface should have a different color.

Applying perturbations to a surface in order to visualize the errors of the estimation process is similar to the point-based method, as both try to suggest uncertainty by rendering a surface that seems randomly rough to the user, but where the absolute roughness value is governed by the error information (Figure 2.13 - left).

Oscillations are a particular type of uncertainty visualization through geometry changes, which due to their temporal nature, fall in the animation category. Even if temporal-dependent error visualization is not a direction our research aims towards, the similarity to previously presented geometrical methods makes it viable for a possible implementation through a time-varying mesh, where the vertices randomly take values on the vertex normals, between the extremes  $P_i$  and  $P_j$ .

The second group of error visualization techniques we want to use and potentially combine with the geometrical one is represented by the attribute addition and/or

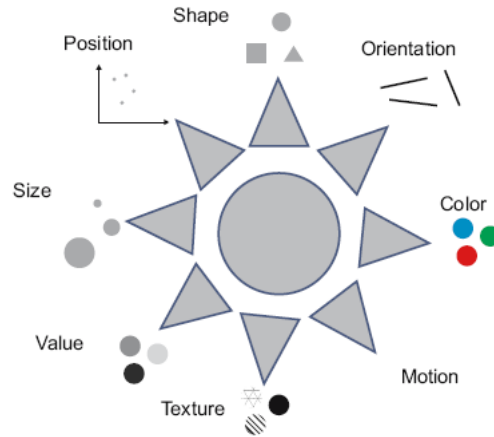


Figure 2.20: Main visual attributes that can be added, modified and combined in object representations

modification. Figure 2.20 contains a chart of the basic visual attributes that can be applied, depending on the scenario and the complexity of the visualization, for uncertainty representation [33]. For each of these variables multiple data-mapping techniques have been developed over the years. We will further emphasize the most important ones to this thesis.

In this segment of variable visual properties, the best known and most widely used methods involve simple color coding and pseudo-coloring (Figure 2.21), which generally have the advantage of generating revealing visualizations that captures uncertainty through the colors of the objects.

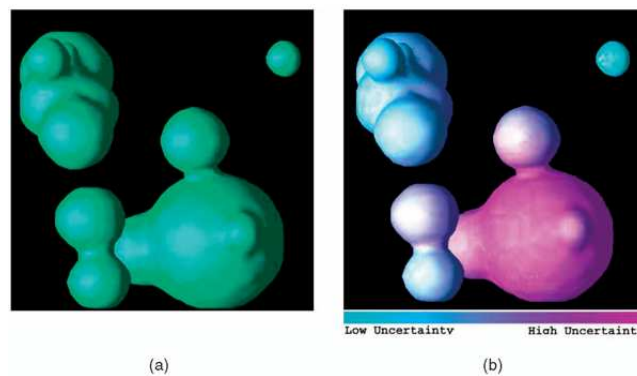


Figure 2.21: Hue-based surface uncertainty visualization - (a) surface without uncertainty (b) surface with uncertainty representation - higher uncertainty is rendered using warmer surface colors

But simple color mapping (in the RGB space) is not the only way to approach such tasks. Changing the values of other attributes like hue (Figure 2.21), saturation, light intensity or luminance (in the HSI space)[23] can improve the rendering results compared to plain color methods. More complex approaches imply the use and variation of shading and lighting, such as altering the diffuse or specular coefficient of the rendered objects (Figure 2.18), or shading variation on curved surfaces.

While color-based methods manage to capture and represent uncertainty in most cases, many times they fail to localize these exactly due to their natural lack of sharpness. To counteract this effect, texture mapping approaches use 2D elements of different shapes and sizes to capture details that color representation could miss. One way of using textures is to populate these with 2D glyphs that would incorporate the uncertainty information in their shape, color or other attribute. An intuitive example for this would be a white surface with red glyphs represented by circles. In areas of high certainty, the radius of the circles would be low so that the glyphs would barely be visible, while in areas of low certainty the size of the circles would increase.

As new methods for visualizing errors by employing geometry attribute modifications we propose the use of transparency and contrast. For the generated surface, each mesh face or mesh vertex would have an opacity attributed to them, proportional to the uncertainty level in that area. This means that for a reconstructed surface, the patches with high certainty would have almost no transparency, while the ones with high possibility to be an erroneous representation the transparency level is increased. The transparency has a manual threshold, as mesh faces are not allowed to reach a completely transparent level.

Contrast in graphics and imaging is determined by the difference in coloring and brightness of two neighboring objects or areas. By making use of this knowledge, we would like to apply a fairly complex texture to the rendered mesh and use contrast as a differentiating feature to highlight areas which contain uncertainty. Parts of the surface that have a low uncertainty value will have a crisp, sharp view of the texture on the surface, while parts with high uncertainty will be represented as strongly blurred. This approach not only has the advantages of color-based methods, in which the uncertainty can be recognized from a viewpoint that is orthogonal to the surface and occlusion is omitted, but also the fact that the human visual system is more sensitive to contrast and color than to saturation and light intensity.

In this section we have presented methods for uncertainty visualization, focusing on surface uncertainty, as well as possible research directions we would like to follow in order to better extract and represent errors for reconstructed surfaces.



## Chapter 3

---

# Real-Time Graphical Methods for 3D Point Cloud Processing

---

The datasets [7] mentioned at the beginning of Chapter 2 contain point clouds that can be structured as two-dimensional arrays. These 2D arrays coincide with the actual structure of the range-image generated by the TOF sensor(s). At each position  $x_1, x_2$ , there is a 3D point, the third dimension being represented by the depth on the  $Z$ -axis, which is the value actually stored at that position. Thus, one has access to both the original neighborhood information and the calculated 3D data for each separate frame. This particular type of point cloud is also entitled range sensor point cloud.

In this thesis, we implement part of the proposed research ideas from the previous subsections (Chapter 2) in C++ and OpenGL [30]. A short list of reasons for using OpenGL as programming interface, and also advantages that would derive from this implementation, includes:

- fast preprocessing/filtering/denoising of the error-prone point cloud data
- advanced visualization possibilities for the uncertainty of the generated multi-resolution patches/meshes, and even for the initial point clouds
- easy integration of OpenGL in various libraries, frameworks and applications. Also, most Robotics libraries are written in C/C++. This would further facilitate the evaluation process of the implemented algorithms
- high flexibility in researching different representations (point sets, patches, planar patches, meshes etc.)
- good documentation on various advanced visualization techniques

Processing speed is a key element in the previous list, as our goal is to generate methods that can be applied in real-time settings. Nevertheless, maybe the major argument towards using a flexible and advanced computer graphics library like OpenGL [60] [6] is suggested by the title of this proposal itself: we need the capability to generate advanced visualization methods that can later be included in various 3D range sensor point cloud processing scenarios.

Matlab scripts are used additionally to OpenGL, especially for the topics that can be investigated separately and without initial time constraints. Such topics are data visualization, preprocessing denoising (Section 2.1) and even surface merging (Section 2.2). While Matlab has the major disadvantage of slow processing, it enables quick development and easy visualization with its vast libraries supporting various fields (including Graphics, Robotics, Machine Learning, etc.) and strong plotting capabilities.

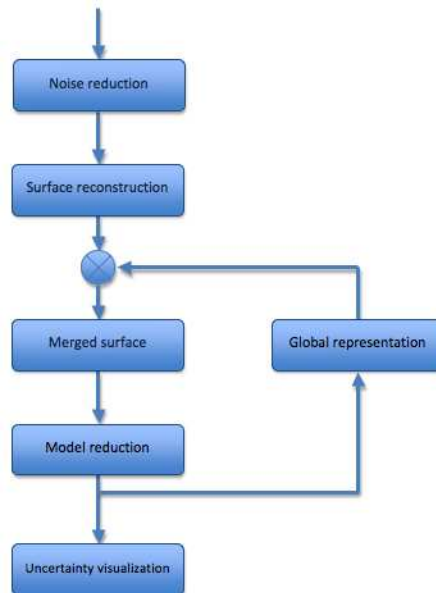


Figure 3.1: Point cloud processing model incorporating the following stages: point cloud noise reduction; surface detection, reconstruction and reduction; merging of point cloud frames - local and global model; global model complexity reduction; global model rendering; uncertainty computation and representation

Figure 3.1 presents our processing model that covers all the steps that are considered for converting the range sensor point cloud into a virtual surface-based representation of the scanned surroundings. Initially, in a step that other approaches fail to cover, we reduce the noise levels present in the point cloud data due to acquisition errors or ambient noise. Next, the points belonging to a surface are selected and used as

support nodes for a mesh representation. In an iterative process, the corresponding surfaces from successive point cloud scans are matched and merged. In order to allow the incorporation of many scans and reduction of computation times, the reconstruction resulting after each fusion is subject of a multi-resolution technique that minimizes the number of required vertices in every mesh. Finally, after the generation of a sufficient global representation, the scene is rendered together with the computed uncertainty levels.

While this thesis considers the entire conversion process to surface representation, it focuses on developing or enhancing only a part of the composite modules; the rest are implemented based on well-known and recognized algorithms from the corresponding scientific field. For the later, references to standard implementations will be supplied and described briefly.

For the implemented algorithms that are part of the processing model and described in detail in the following sections, we achieved our goal of online computation, as the cumulative execution time for all the stages varies between 0.1 – 0.25 seconds. In the worst case, these values still allow for a processing of 4 point clouds frames per minute, sufficient for most online tasks. Moreover, over half of the run-time is attributed to the preprocessing step, which is still implemented in Matlab.

To allow later performance comparisons, we mention that the Matlab and C++/OpenGL based applications that are part of this thesis were executed on a computer equipped with a 2.4 GHz Intel Core 2 Duo processor, 2 GB of RAM and an Intel 965 video card.

The following sections highlight the research and implementation of a set of methods, representing various stages of the procedural model depicted in Figure 3.1.

### 3.1 Error Reduction with Anisotropic Diffusion and Wavelet Denoising

Most of the methods dealing with 3D range point clouds and with the extraction of surfaces do not consider the presence of noise in the scan datasets or try to overcome this problem by using noise insensitive techniques for the conversion. While methods that are immune to error prone data do produce good results, they usually require complex computations to be executed and thus represent limited usefulness for tasks that require online execution. Also, there are complex 3D based techniques that attempt filtering the point cloud in order to remove the acquisition noise (Figure 2.15) as described in Chapter 2.1, but these too are rather expensive in terms of processing speed. As a result, in most real-time implementations the point cloud is not denoised and simplified noise insensitive approaches are used.

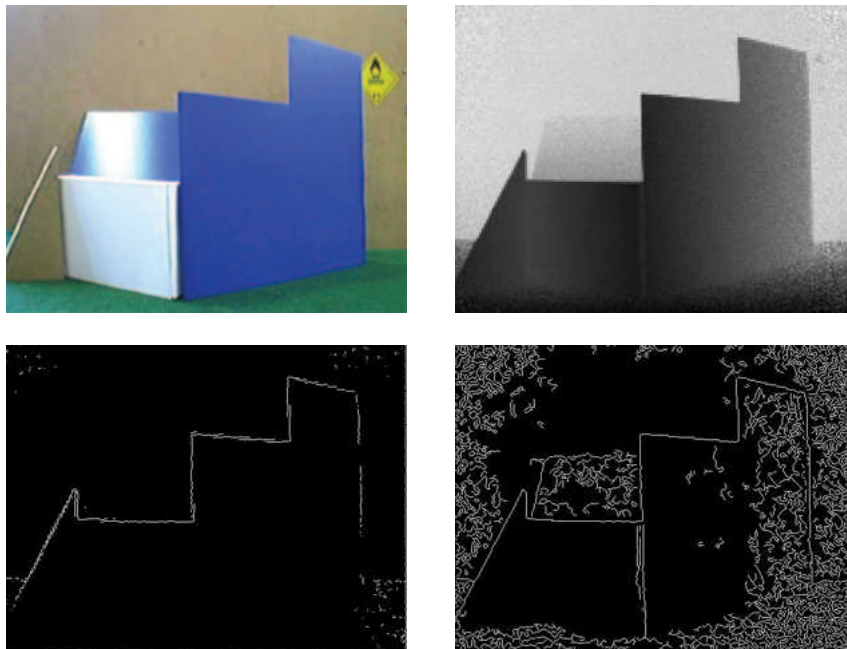


Figure 3.2: Scene photography and normalized grayscale distance image obtained with the SwissRanger SR-3000 (top), edge detection on original distance image using Prewitt (left-bottom) and Canny (right-bottom)

Figure 3.2 presents the problems of noise prone 3D range sensor point clouds for the case of SwissRange 2D mapped and normalized distance images. Even though the photographical representation of the scene is quite intuitive, the grayscale distance image loses a lot of the spatial information due to noise introduced by the sensor upon the surfaces (i.e. the pixelated areas in the corners of the distance image) and due to ambient noise (i.e. external lighting that contains IR light).



To avoid the negative influence of the noise on the analysis of the point cloud data, we propose two methods for eliminating it - *anisotropic diffusion* and *wavelet denoising* with phase preservation. The idea to which both algorithms adhere, is to reduce the noise levels inherent on all surfaces and maintain the edge features, as boundaries are used in many approaches as vital information for correct segmentation of the 3D points into surfaces. Furthermore, we are gaining the advantage of faster processing, as these methods analyze 2D grid structures and not a complete 3D space, and therefore use the particularity of grid structured range images.

The used *anisotropic diffusion* filter is based on the algorithm of Perona and Malik, as detailed in [44]. The basic idea of the approach considers the summing of the original image with a set of derived ones, obtained by convoluting the base image with a Gaussian kernel. The kernel should have a variance equal to  $t$ , in order to make it space invariant, while the set of derived images can be viewed as the solution to heat conduction in Physics.

Our implementation uses the normalized grayscale distance images as input for the diffusion filter and follows these main steps:

- Use the original distance image  $I$  to create a filter  $I_f$  by adding a border of zeros to it
- Compute the differences  $\delta_i$  between  $I_f$  and  $I$  shifted by one pixel in the four main directions
- Compute the conduction  $F_c(i)$  in all directions
- Compute the sought diffused image as:

$$I_{diff} = I + \lambda \cdot \sum_{i=1}^4 F_c(i) \cdot \delta_i \quad (3.1)$$

Where  $\lambda$  controls the speed of diffusion and  $F_c(i)$  is the value of the conduction function computed in the  $i^{th}$  direction.

As suggested in [44], different conduction functions can have better results depending on the task. In order to maximize our chances for obtaining a good diffusion filter, two conduction functions are considered:

$$F_c = e^{-\left(\frac{\delta}{k}\right)^2} \quad (3.2)$$

and

$$F_c = \frac{1}{1 + \left(\frac{\delta}{k}\right)^2}; \quad (3.3)$$

In this case, the parameter  $k$  controls conduction as a function of gradient. If  $k$  is low, small intensity gradients are able to block conduction and diffusion across step edges; on the other hand, a large  $k$  reduces the influence of intensity gradients on conduction.

While the meaning and the impact of the parameters is quite intuitive, selecting the right parameters for the filter constitutes a crucial role in the production of a good model. Our experiments have shown that noise reduction with maximal edge preservation can be achieved with a lower value for  $k$  ( $k=20$ ) to block diffusion over edges and small  $\lambda$  ( $\lambda=0.1$ ) for an increased stability (or reduced diffusion speed), in the case of edge methods like Prewitt, Sobel and Roberts.

As edge detection using the Canny method follows very different and more complex guidelines, the best results for it have been obtained with a slightly different setup: a high diffusion speed of  $\lambda$  ( $\lambda = 0.3$ ) and the same  $k$  ( $k=20$ ) for no diffusion over edges. Also, the filtering was executed up to three times and the effect convoluted. All these positive results have been achieved by employing the conduction function 3.2.

To further limit the amount of artifacts present in the edge images, we introduced two additional filtering steps, this time oriented towards the reduction of noise or false positives directly in the edge extraction. While edge detection and improvement are not the direct topic of this work, the capability of correct detection of edges in the distance image is considered a stable measure for the effects of an error reduction filter. The reasoning is that even if the diffused, or more generally filtered image, is not perfectly purified, by applying a set of easy and computationally inexpensive methods, we can further improve the results.

Two of the problems present on some of the diffusion filtered edge images are the presence of isolated pixel edges and false boundaries detected around the margins of the image due to the way anisotropic diffusion convolutes the original and derived images. To overcome these issues, the edge image, which is basically a binary image, is rapidly post-processed by removing any edges closer than two pixels to the image boundaries and any simple pixel edges that do not present the possibility of continuity. More clearly, if a pixel is considered on an edge, but none of the 8 neighbors is a border, it will not be automatically removed and the second order neighbors are checked to see if there is a continuity (i.e. if the first pixel to the left and right are not borders but the second ones to the left and right of the current edge pixel are, then the edge pixel is not removed).

The best edge extraction solutions after anisotropic diffusion, border clearing and

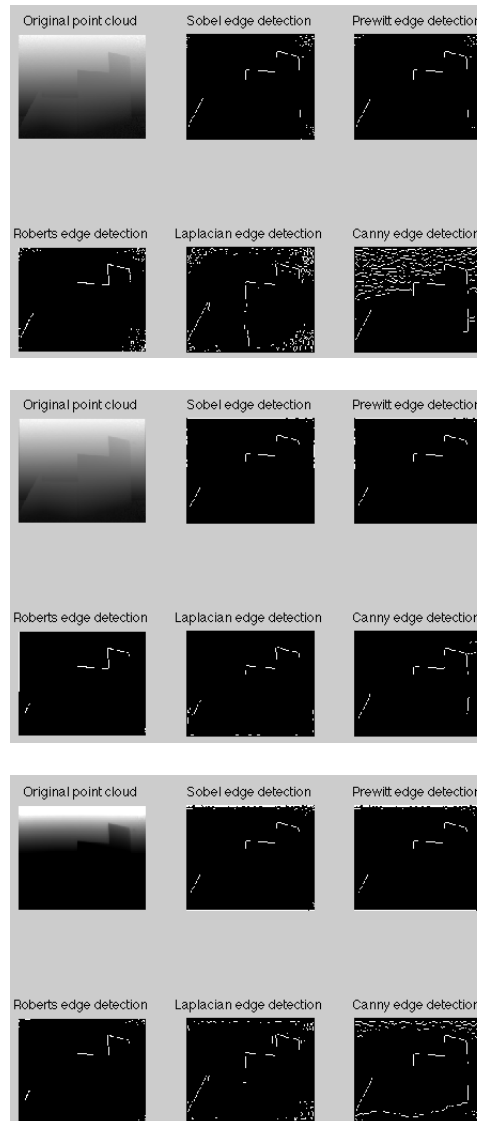


Figure 3.3: Edge detection algorithm results on original distance image (top). Best edge detection results on distance images filtered by anisotropic diffusion (middle). Best edge detection results on distance images filtered by wavelet denoising (bottom).

single pixel removal can be inspected in Figure 3.3b. Moreover, a comparison between the diffusion based edge extraction and the same methods applied on the raw distance images shows an improvement in the sense of reduction of false positives, as well as a conservation of the real edges. An important fact to consider is that the edge detection methods that were used are constant over all the experiments presented in this thesis, and use an automatic parameter calibration where required.

Although Figure 3.3 presents only one distance image, our results have shown that an efficient noise reduction and thus edge extraction can be achieved for over 90% of the tested point clouds.

In terms of run-time, the execution of anisotropic diffusion takes up to 0.02 seconds on a single range point cloud. Even if for Canny edge detection better results can be obtained with up to three consecutive filtering operations, 0.06 seconds is more than enough for an online execution, not to mention that a speedup is definitely expected for a C++ implementation, over the current Matlab one.

Compared to anisotropic diffusion, wavelet denoising is a fairly more complex, but also a more promising filtering method, as suggested in [36]. The concept that guides denoising methods is that the initial space, which in our case is represented by the 2D distance image, has to be transformed into another domain where the noise can be easily identified and eliminated. After this removal, the remaining information is morphed back into the initial space, thus reconstructing a noise-free image.

Wavelet denoising relies on the fact that an image is basically a combination of wavelets with a compact support. While this support can be described by wavelet coefficients that are highly localized, the coefficients corresponding to noise signals are rather distributed. Following this line of thinking, the coefficients that support the main information would be more easily detectable.

Our approach reimplements the denoising method with phase preservation as described in [36]. The technique uses non-orthogonal, complex valued, log-Gabor wavelets instead of the more commonly encountered orthogonal or bi-orthogonal wavelets, and has the additional advantage of allowing the automatic computation of a noise threshold value at each scale. Also, in order to preserve the phase information of our distance image, the local amplitude and phase have to be extracted at every pixel. This is achieved by using wavelets that are pairs of symmetric and anti-symmetric, as well as by employing a wavelet transform.

At the heart of the approach lies the convolution of the signal and each of the quadrature pair of wavelets. To obtain this, one needs to compute the frequency component of the signal, which can be described as a complex value:

$$C = E_n + i \cdot O_n \quad (3.4)$$

Where  $E_n$  and  $O_n$  are obtained from:

$$\begin{aligned} E_n(x) &= I(x) * W_n^E \\ O_n(x) &= I(x) * W_n^O \end{aligned} \quad (3.5)$$

In the previous equation,  $I$  denotes the signal, while  $W_n^E$  and  $W_n^O$  represent the even and odd-symmetric wavelets for a certain scale  $n$ .

By further computation, one can extract the amplitude and phase of the transform at a given scale, as:

$$A_n(x) = \sqrt{E_n(x)^2 + O_n(x)^2} \quad (3.6)$$

And

$$\Theta_n(x) = \text{atan2}(O_n(x), E_n(x)) \quad (3.7)$$

The response vectors extracted by this procedure for each  $x$  and at every level are the basis for the localized representation. In this representation, the denoising procedure consists of noise threshold deduction for each scale and reduction of the response vector amplitudes accordingly.

From an implementational point of view, the algorithm follows these steps:

- Compute the Fourier transform  $I_{FTT}$  of the initial distance image
- As described above, compute  $E_n$  and  $O_n$  as two matrices: the elements of  $E_n$  have values equal to the  $x$  coordinate relative to the center, and ones of  $O_n$  have values equal to the  $y$  coordinate relative to the center
- Calculate the amplitude and phase of the transformation, as described in Equation 3.6 and 3.7
- Initialize the total energy  $ET$  that will retain the energy levels from different scales
- For each orientation:
  - Compute the angle of the filter in the current orientation
  - Based on the filter angle and the phase, compute the angular distance  $\Delta\Theta$  from the specified filter orientation for each point
  - Compute the angular filter component as:

$$Spread = e^{\frac{-\Delta\Theta^2}{2\Theta_\sigma}} \quad (3.8)$$

Where  $\Theta_\sigma$  is the standard deviation of the angular Gaussian function used to construct filters in the frequency plane

- For each scale:
  - \* Compute the log Gabor function, which represents the radial filter component, and multiply it with the previously calculated angular filter to obtain the filter for the current scale

- \* Convolve the transformed image  $I_{FFT}$  with the even and odd filters to extract the current energy  $E_i$
- \* If the current scale is the smallest, estimate the mean and variance in the amplitude response of the smallest scale filter pair at this orientation
- \* If the current scale is not the smallest, use estimate of the noise amplitude distribution for the smallest scale to compute the one of the current scale. This is achieved by using the property that the amplitude response is directly proportional to the filter centre frequency in 2D images
- \* Based on the estimated noise response distribution at each scale, set the noise reduction threshold to be  $T$ , where  $T$  is higher by  $k$  times the standard deviation than the mean of the distribution

$$T = \mu_{Rayleigh} + k \cdot \sigma_{Rayleigh} \quad (3.9)$$

- \* Add the current energy  $E_i$  to the total one  $ET$  and calculate the wavelength of the next filter

Additionally to using the algorithm, one has to be able to determine the right values for the input parameters, from which many relate to the specification of the filters in the frequency plane. The parameters that were iteratively set for tuning the denoising approach to the type of error inherent to the point cloud data, were:

- $NSD$  - number of standard deviations of noise to be filtered
- $NS$  - number of filter scales
- $MF$  - multiplication factor between scales
- $NO$  - number of orientations

In the sense of highest noise reduction with best edge preservation when applying an edge detector, the best denoising results for Prewitt, Sobel and Roberts were obtained with a filter setup that considered only 2 standard deviations  $NSD$  for the rejected noise, but had a high number of filtered scales ( $NS=7$ ), a powerful multiplication factor ( $MF=3$ ) and considered 8 different orientations  $NO$ .

Again, to obtain similar results for the Canny edge detection, the parameters had to be reevaluated. The standard deviations number  $NSD$  was increased to 5, only 5 filter scales  $NS$  were used with a multiplication of 2 between them and 6 different orientations  $NO$  considered.

These two configurations for the denoising algorithm performed generally well, reducing false positives for the edge images and in some cases, even allowing the detection

of previously unseen edges! An example for the results of wavelet denoising with phase preservation is given in Figure 3.3c.

The execution time of the denoising filter varies between 0.07 and 0.1 seconds, which is enough to accommodate real-time preprocessing of distance scans from a mobile robot.

As described above, one of the metrics employed for estimating the performance of the noise filtering is given by the edge detection images that are generated from the already preprocessed images (Figure 3.3). Nonetheless, in order to avoid using a single performance metric another way of evaluating the error filtered data is proposed. As using only distance information or positioning for 3D point clouds is rarely enough, an intuitive way of detecting quasi-planar surfaces relies on the computation and evaluation of the normals.

Mathematically, a planar segment should have the same value for the normal at every point of the surface. Various established methods use this property and compute the normals at every vertex in order to add another degree of detection and separation for the later segmentation process.

As a secondary measure to evaluate the precision and performance of the proposed filters, the previously stated property and the surface normals are used. More exactly, the set of 3D points composing the cloud is translated into another 3D space, where the axis are defined by the projection of the normal at point  $x$  on the three original axis  $X$ ,  $Y$  and  $Z$ , weighted by the distance  $D$  of the point to the sensor. In this new 3D space, a noiseless scan of a planar surface should result in a set of points that all have the same normal direction, thus the same normal projections.

This means that these points would be grouped and that the only variation would be introduced by the distance  $D$ . More generally, in a perfect scenario, if multiple surfaces are part of a point cloud set, all the points corresponding to the same surface would be clustered inside the new space and the clusters would be (in most cases) clearly separated. The problem is that in a real, noise prone scan, the information is distorted and the clusters are no longer clearly distinguishable or/and many points can not be clearly attributed.

Figure 3.4 presents the typical results obtained by the transformation in the normal based 3D space, in the case of the original point cloud, as well as for the two filtered versions. Notice that in both cases, especially the wavelet denoising, the desired effects of compressing the clusters, eliminating non-attributed points and maintaining the distance between clusters for classification purposes is achieved. The visualization

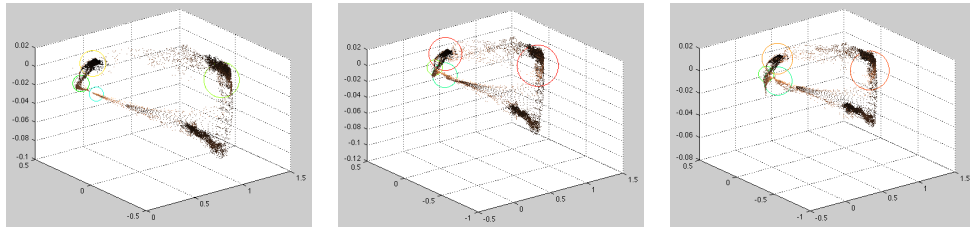


Figure 3.4: Comparison of noise reduction in  $N \times D$  space based on original image (left), best anisotropic diffusion filtered image (middle), best wavelet denoised image (right).

is not only important for the evaluation of the filters, but also for the segmentation of the point cloud (presented in Figure 3.5), such that subsets of points that are known to embody elements on the same quasi-planar surface are selected. The notion of quasi-planar surface is used, as even after error filtering there can still be points that lead to a noisy planar surface; nonetheless, the elements of such a patch need to be accepted by the method.

The clustering of corresponding 3D points inside the  $N \times D$  space is also used as the base for segmentation (Figure 3.5), in order to allow the later reconstruction of a mesh. The segmentation process is not a topic of this thesis, its implementation is rather brute, but has the distinct plus of completing the pipeline and allowing for more resources to be dedicated to the actual topics. The clustering method itself is based on the idea of surfaces being composed of closely situated points that for a tight group, and therefore can be identified by employing a mixture of Gaussians.

Initially, it is considered that each point in the  $N \times D$  space represents the center of a Gaussian. Then, in an iterative process, all the points that are inside another Gaussian are identified, resulting in the merging of the two clusters and updating the cluster center. This continues while there are still clusters that can be merged. In the end, a threshold establishes which Gaussians actually include the points of a surface by setting a lower limit to the number of 3D points required. In some cases, when the segmentation was insufficiently accurate, manual adjustment was added to return a more appropriate set of points that describe the surface.

During our evaluation of the edge detection results on the filtered distance images, besides the positive outcome with the elimination of false positives, preservation of edges and in some cases extraction of new edges (i.e. based on phase preservation), it was observed that applying various edge detection methods on the two sets of noise reduced images eliminated errors rather consistently, but generated different correct edges.



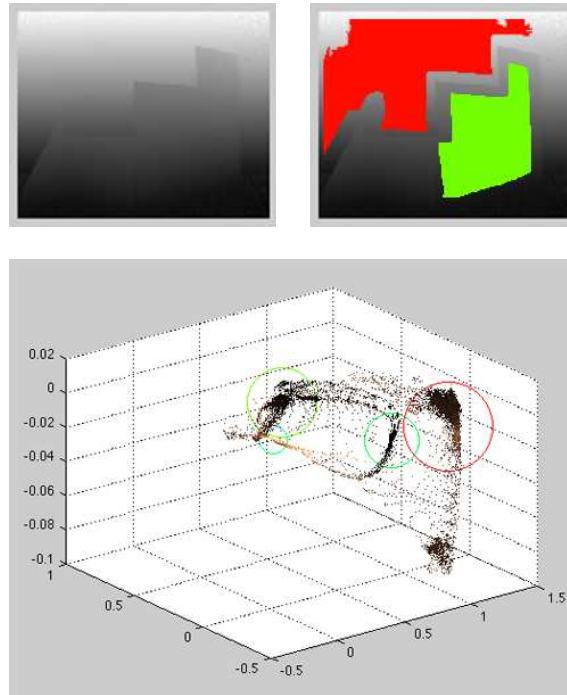


Figure 3.5: Normalized distance image (top-left). Corresponding 3D space representation of initial point cloud after  $N \times D$  transformation (bottom); the colored circles highlight the detected clusters of 3D points part of the same surface and its size. Initial normalized distance image with an overlay of the two largest correctly detected clusters/surfaces (top-right); surface color corresponds to cluster color in the previous subfigure.

In order to use this advantage, an approach was devised to reconstruct as many possible edges from the various filtered images and sumate them into one edge image. Two examples are presented in Figures 3.6 and 3.7, and the basic idea for implementing the method follows these steps:

- Apply anisotropic diffusion with the second set of parameters on the original distance image
- Compute the Canny edge detection image of the diffused image
- Apply the boundary and single pixel filter (previously described) on the edge image and store the result in  $E1$
- Restart the process from the original image and apply denoising with the second set of parameters on it
- Use the denoised image as input for an anisotropic diffusion step with the first set of parameters

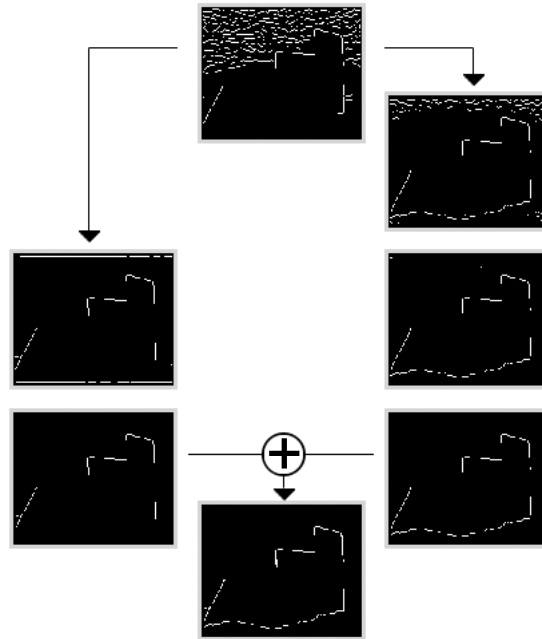


Figure 3.6: An error reduction, as well as an edge preservation and enhancement scheme for 3D range point clouds. The sample point cloud contains 6 surfaces and all images use Canny edge detection. Top image - edge detection on original distance image. Left branch - denoising (1) followed by border pixels removal and single pixel removal. Right branch - denoising (2) followed by anisotropic diffusion for error removal and border pixels removal and single pixel removal. Bottom image - binary sum of the edge detection images.

- Apply the boundary and single pixel filter to the previous result and store the edge image in  $E2$
- Convolve  $E1$  and  $E2$  to obtain the final edge image

Although this portion of the research is not directly related to the initially proposed one, the positive results obtained based on the convolution of multiple edge images obtained from filtering variations are impressive and further support the need for a preprocessing step for noise reduction in the case of sensor datasets.

Most of all, the technique is still compatible for an online execution, as the total run-time does not surpass 0.2 seconds, thus allowing for 5 fps to be analyzed. While this can seem a small value, especially as the main processing steps still need to be executed, code optimization should reduce this initial model stage.

Once the sets of points are extracted as the based for surfaces inside a point cloud, this information needs to be represented. Normally, the rendering of the global scene

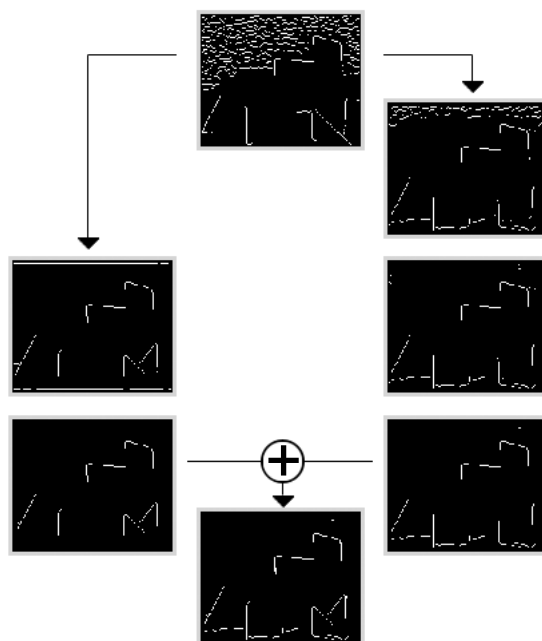


Figure 3.7: An error reduction, as well as an edge preservation and enhancement scheme for 3D range point clouds. The sample point cloud contains 6 surfaces and all images use Canny edge detection. Top image - edge detection on original distance image. Left branch - denoising (1) followed by border pixels removal and single pixel removal. Right branch - denoising (2) followed by anisotropic diffusion for error removal and border pixels removal and single pixel removal. Bottom image - binary sum of the edge detection images.

(Figure 3.1) would be executed at the end of the pipeline, but we present this issue at the one-mesh level, for a better perception of the problem.

To display a mesh based on a set of 3D scattered points, the Delaunay triangulation [34][15] scheme is implemented. More precisely, to achieve better performance and simplify the task, a 2D triangulation is used based on [18] and the depth information of the point cloud is disregarded, as we are basically dealing with a 2D patch.

In this approach, the mesh representation generated from the surface points considers selecting a point approximately at the middle of the grid and creating an edge with the nearest neighbor. Then, the right side of the edge is scanned and the maximal angle point is selected. These three points are used to compute a circle - if the circle has any other points in it, the process is restarted with the new point, otherwise the three points can be triangulated and stored as a triangle configuration. By computing all the available triangles that have this property of no other point being in the surrounding circle, a triangular mesh can be generated. This 2D mesh

can then be extended by simply adding the  $z$  dimension to each point and rendering in 3D space.

Until this point of the procedural model (Figure 3.1) we have implemented noise reduction, segmentation and mesh rendering. The next section will detail our approach for surface matching, merging and multi-resolution.

## 3.2 Surface Multiresolution Representation

As the information from 3D point cloud frames is added to the global reconstruction of the scanned surroundings, the need of a method for handling the large number of surfaces and support points becomes increasingly obvious. To overcome this problem, we propose a multi-resolution approach particularly tuned for almost 2D open mesh surfaces generated by range sensors.

Applying a multi-resolution or subsampling solution for surfaces from a single point cloud or from a global representation can actually be reduced to the problem of varying the number of nodes inside a single mesh. We then estimate the distance of each mesh to the viewport in order to decide on the number of vertices to be used, or LOD to be displayed, by means of a linear function.

To understand the procedure, one has to consider that the initial mesh representation of a surface uses all the vertices of the 3D point cloud that were previously selected as part of the same surface. A multi-resolution approach would thereby imply that from the list of selected support nodes of the mesh, we remove the least important ones, iteratively. By doing this, the mesh is gradually being simplified, and thus its rendering becomes less expensive. Additionally, maintaining the information-packed nodes in the lower resolution representations maximizes the conservation of support features, and therefore the overall shape of the mesh.

In order to implement this, we need to create an evaluator that would quantify the information content of each vertex to achieve a correct representation of the surface. Ideally, sharp features like corners or bumps would get a higher value, while planar areas - a value close to zero. In this manner, we should manage to preserve the overall shape of the surface even at low resolutions.

Intuitively, in a scenario with quasi-planar surfaces, a majority of points should receive values close to zero, as removing a support situated inside an almost flat patch should not affect its boundaries. An exception to this is the case when the vertex is displaced on the 3<sup>rd</sup> axis - depth of the sensor data. While this displacement might be due to the remaining error after preprocess filtering, it could be also the case that we are dealing with a real sharp feature on the surface.

To accommodate all these ideas, we introduce an algorithm that evaluates the importance of a node  $X$  inside a mesh based on the four closest neighbors  $X_1 - X_4$ . These four neighbors are used to estimate the value  $z_e$  of the node  $X$ , resulting in  $X_e = (x, y, z_e)$ . The difference between the actual value  $z$  of  $X$  and the estimated one  $z_e$  then represents the quantity of information encoded in that vertex.

In the initial setup, where the points of a surface are successive on  $x$  and  $y$ , the four neighbors are simply the closest pixels in 2D, that is the ones above, below and on the sides of the pixel, as suggested in Figure 2.4. The depth or  $z$  component is neglected, as this approach focuses on retrieving the depth displacement features. Therefore, limiting the search and comparison space to 2D, another opportunity to make use of the point cloud grid structure is captured.

As the method advances and points are being removed, the mesh vertices need to be reevaluated, in order to get a new estimation of their value. As a natural generalization of four direction look-around, the introduction of a dynamical computation of the node importance is based on the idea that through vertex removal, remaining nodes can incorporate some of the information of non-displayed nodes and increase their sharpness value. In the case of missing pixel neighbors due to positioning on the mesh boundary or previous removal, the method is generalized as presented and uses the four values of the closest nodes to the current  $X$ .

The multi-resolution concept is implemented by adding a dual vector of vertex indexes and corresponding vertex significance for each mesh. The vector is then sorted based on the node importance and a pointer to the current LOD is stored. The displacement of the LOD pointer allows thus for the rendering of all resolutions, from the coarsest to the finest, which contains every selected surface points. Moreover, after each change in resolution, the importance of the vertex is recomputed for the "visible" part of the mesh.

We know that the computation of the estimated value of  $X$  is based on the four closest neighbors, but we still need to clearly specify a function that gets the neighbors as input and outputs the estimation. To go into further detail, we implement two 2D interpolation schemes in order to estimate  $z_e$ : inverse distance weighting and natural neighbor interpolation.

*Inverse distance weighting* (IDW) or the Shepard interpolation is a well-established method for a rapid interpolation in 2D, as described in [1]. The corresponding function that guides the computation of the local estimation is a particularization of the Shepard function and can be described as:

$$z_e = \frac{\sum_{i=1}^4 \frac{1}{d(x,x_i)} \cdot z_i}{\sum_{i=1}^4 \frac{1}{d(x,x_i)}} \quad (3.10)$$

Where  $X$  represents any given point,  $X_i$  is one of the four known points,  $d$  is the metric operator or the distance between  $X$  and each  $X_i$ . You can notice that as the weight decreases, the distance increases from the interpolated points. The additional advantage of this function is that if the closest known points are coplanar, the estimation will be coplanar too. Therefore, the difference  $(X - X_e)$  is very likely to

capture a distortion in the surface, if present.

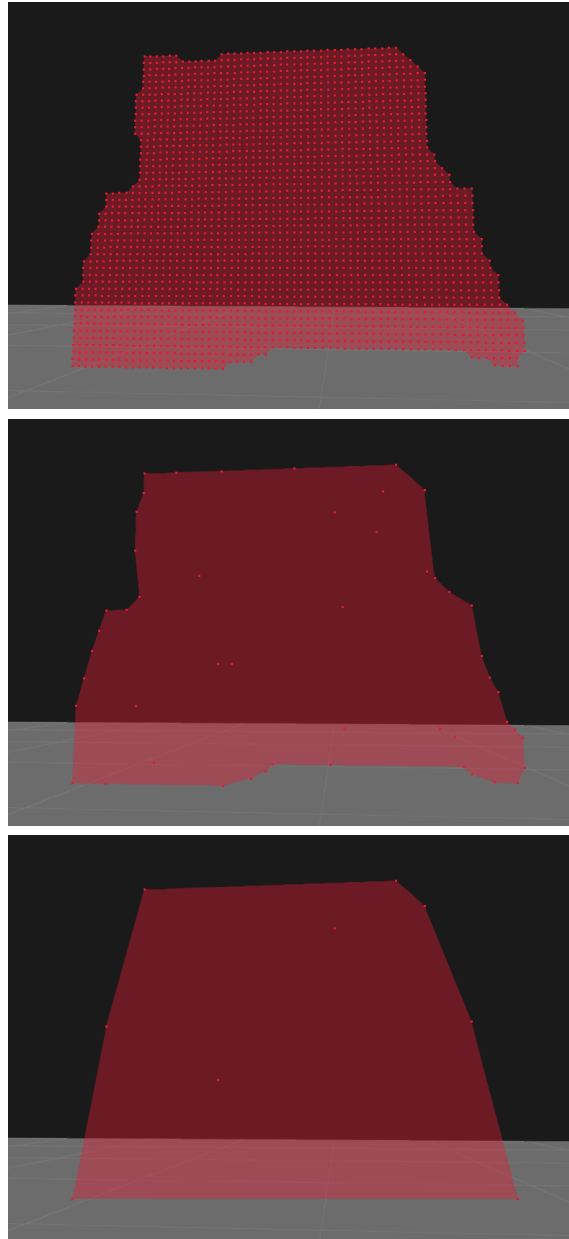


Figure 3.8: Multi-resolution representation of a surface mesh from fine (top) to coarse (bottom) resolution. The usefulness of a vertex is computed by comparison with the Shepard interpolated value. The natural neighbor interpolation obtains similar results.

The second spatial interpolation function employed is given by the *natural neighbor interpolation*, also known as the *Sibson interpolation* [43]. The method is based on

the Voronoi tessellation and the Delaunay triangulation on a discrete set of 2D points, shortly presented in the previous chapter.

The main idea is that for each point  $X$  which we want to evaluate, we add  $X$  as a site to the Voronoi diagram of the original sites, and average the values of the sites weighted by the fraction of the cell for  $X$  previously covered by each other cell. This relatively complicated procedure results in a continuous function, smooth everywhere except at the original sites.

In mathematical terms, the function used to estimate the value of  $z_e$  is:

$$z_e = \frac{\sum_{i=1}^4 A_i \cdot z_i}{\sum_{i=1}^4 z_i} \quad (3.11)$$

Where  $z_i$  is the value of each known neighbor and  $A_i$  is the intersection of the  $X_i$  neighbors Voronoi area with the area of  $X_e$ , when  $X_e$  is introduced as a known point and the 2D surface is retessellated.

While the Sibson approach is computationally expensive, as multiple surface areas have to be calculated, it has the advantage of being a high precision method. Also, as we have already used the Delaunay triangulation to represent the reconstructed surfaces (Chapter 3.1), we consider that most of the structural and computational complexity is already embedded in the procedural model, and therefore using this interpolation scheme can be considered a reasonable investment of resources.

A representation of the multi-resolution implementation for a surface, based on the Shepard interpolation function is presented in Figure 3.8. The results obtained with the natural neighbor interpolation function are quite similar, suggesting that the Shepard interpolation, which is less computationally expensive, should be appropriate in most cases for the importance estimation.

Capturing the sharp features along the depth axis of the 3D point clouds is only half of the solution. In the case of non-closed surface representation, a very important factor is the preservation of boundaries or edges. Even if the surface boundary can be error prone, a conservation of the overall shape is desired, for as many LODs as possible.

To include border preservation in the proposed approach, an algorithm is used that increases the significance of the marginal points, by adding to the values computed with the interpolation schemes. To maintain consistency, the methods for sharp edges consider the same strategy: remove the current node, estimate the local boundary based on the neighbors and compute the distance between the actual pixel and the new margin. Again, the computations use the advantages of a 2D grid representation.



The stages of the method for surface boundary preservation, as implemented in this thesis, are:

- Mark the boundary points at the finest resolution. A border point is one that has at least one neighbor missing
- Create a two-way list of the boundary pixels in order to be able to search the neighborhood of a current point
- For each visible boundary element, compute the distance to the segment created by the connection of the immediate neighbors
- Add this distance to the depth vector at the corresponding position

The last step ensures that edge vertices are maintained as much as possible in every LOD, by increasing their importance, as these might represent sharp features on both  $z$  and  $(x, y)$ .

In terms of performance, the multi-resolution approach is sufficiently fast to maintain real-time execution and high enough frame rate, as tested for up to 10 surface meshes.

### 3.3 Mesh Matching and Merging

After determining how surfaces are handled based on the distance to the view screen, the next step covers one of the main stages of the processing model: the mesh merging. In order to be able to fuse two meshes generated based on different point clouds, we first need to establish the correspondence of any two such sub-surfaces. As this is something that has been widely covered in the literature and good results have been obtained with various methods, we concentrated our efforts on the actual merging process. Nevertheless, a matching step is required.

Therefore, the *iterative closes point* (ICP) algorithm [3][63] is implemented to minimize the difference between two point clouds or, in our case, the distance between the point sets that have been selected as part of a surface. The technique basically follows these steps:

- Associate points by the nearest neighbor criteria
- Estimate transformation parameters using a mean square cost function
- Transform the points using the estimated parameters
- Re-associate the points and restart the process from the first step

The result in our case is not only a transformation that guides the positioning of the second point cloud data relatively to the first one, but through the comparison of the distance between the sub-point clouds represented by the surface points, the correspondence of two sets of points is established, if present. As we employ the basic algorithm for ICP, matching errors might occur; these are currently overcome by manual adjustments and matching, as more precise and complex algorithms for matching can be introduced at a later time. The odometry of the robots that have the sensors mounted is not used, as this is quite noisy, and exceeds the scope of this thesis.

Once it is known which two sets of points correspond to each other, if any, the merging algorithm is executed. The technique uses a rather simple approach and has the added advantage of maintaining the structure that allows multi-resolution to work on the merged mesh.

Based on the ICP matching of the two sets of vertices, we can fuse the sets of points by computing an average for each point that has a close enough correspondence in the other set. These new averaged vertices are then stored in a vector, as was the case for any surface points. Additionally, the points from both sub-clouds that had no correspondence are added to the end of the vector.

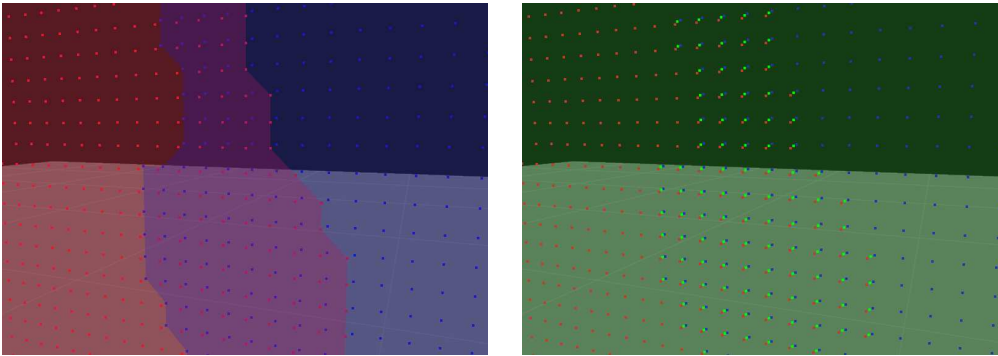


Figure 3.9: Merging two corresponding mesh surfaces - rendering of two corresponding meshes separately (left) and rendering of the resulting merged mesh (right). Vertices present in both meshes are averaged and presented as green support points, while others are simply added to the next global surface.

An alternative, simpler implementation, considers only adding the two sets of points inside the same vector, thus eliminating ICP related computation. But this has the disadvantage that the number of nodes supporting the finest resolution representation is much higher, which can become a problem after adding many point clouds. The new vector of 3D points is then displayed and triangulated using the Delaunay method as described in the Chapter 3.1. Figure 3.9 presents the merging of two sets of 3D points to create a new, extended surface rendering.

### 3.4 Uncertainty Visualization Methods for Surfaces

Figure 3.1 presents the uncertainty visualization as the last stage of the conversion process from 3D point cloud to a list of rendered surfaces. While it is at the end of the pipeline and also not a mandatory module to be executed for each point cloud to be processed and added to the global representation, it carries a high importance to the correct and complete representation of the extracted surfaces.

As one of the main topics of this thesis, error visualization has been already broadly discussed in Section 2.4, where various general uncertainty representations have been presented and analysed. In the following paragraphs, we present a set of techniques for graphically representing uncertainty for error prone mesh surfaces generated based on point cloud data. This can be of particular importance when submitting virtual environments that have been reconstructed from a series of 3D point clouds (e.g. successive scans of a range sensor mounted on a mobile robot) to human analysis of the global representation or local feature particularities.

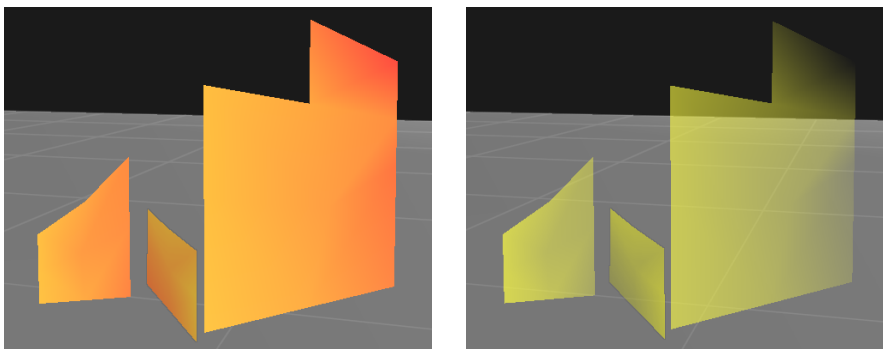


Figure 3.10: Implemented techniques for visualizing the multi-resolution uncertainty of the resulted surface based on color mapping and transparency.

A first group of methods focuses on modifying preexisting geometric attributes, in this case the color and transparency levels of the surfaces. The uncertainty is then represented by adding color or transparency mapping to the meshes, as presented in Figures 3.10a and 3.10b, thus encoding the accuracy of the displayed mesh in those attributes.

While the geometry attribute related techniques encode the error of the surface, are quite inexpensive and allow for a good human perception of variations in intensity levels, they offer the downside that the information is not quantitatively proportional to the geometry. Human perception has a better capacity of comparing similar types of visual information. Therefore, as a surface is first of all geometrical, the uncertainty in shape and position is better transmitted by the use of added geometry.

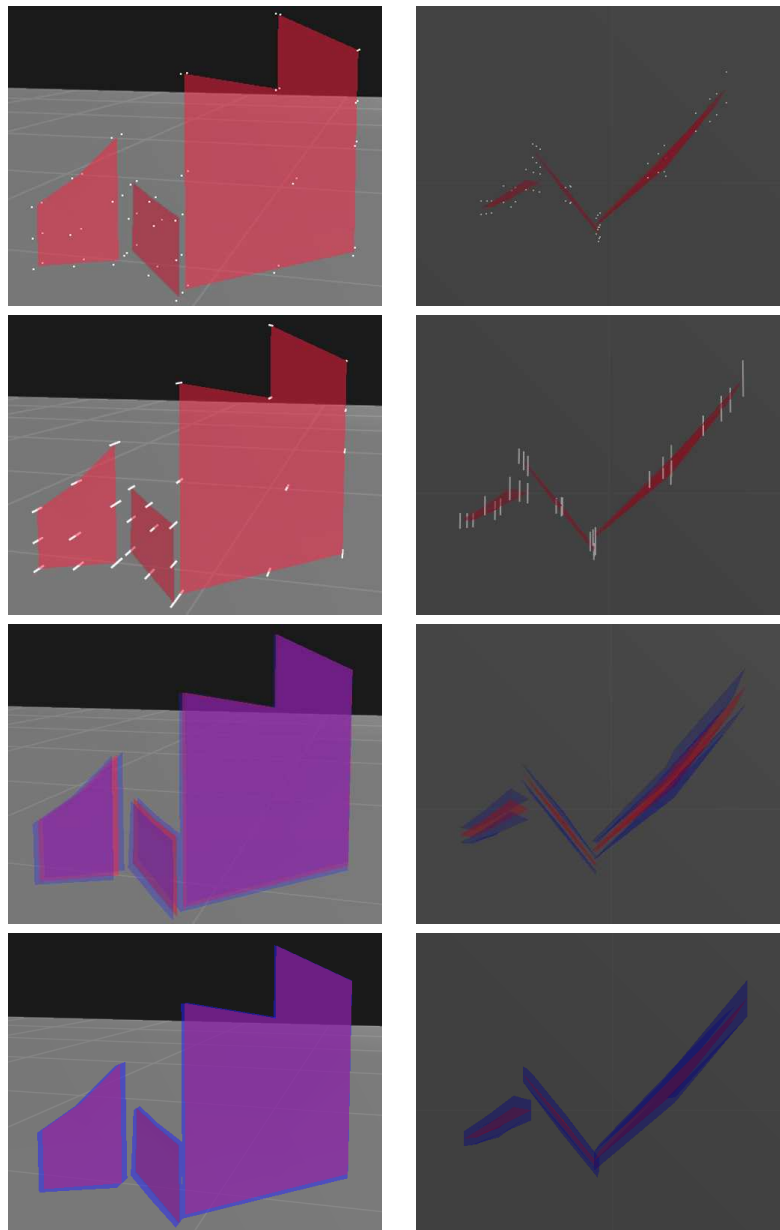


Figure 3.11: Implemented techniques for visualizing the multi-resolution uncertainty of the resulted surface based on geometry addition (top to bottom): points, bars, uncertainty boundary and uncertainty volume.

To satisfy this, Figure 3.11 shows implementations of surface uncertainty representations that rely on methods highlighted in the previous chapter - points, bars, meshes. In order to render the additional geometry, the normals of the surfaces at each mesh vertex are computed. As detailed in Algorithm 1, the geometric elements are then

positioned on both sides of the mesh, on the normal to the corresponding node. These two mirrored values and the mesh vertex are then used to graphically display the uncertainty at that particular point by different geometric structures. The usage of points, as well as the one of lines to depict uncertainty has the advantage of minimal computation and a good communication of the global information. On the other hand, arranged sets of points and/or lines are not too common to the human environment.

However, as these techniques solve our direct comparison problem and give the user the possibility of perceiving the covered probability space of each surface, problems might still arise when they are used at a high LOD. In these cases, the mesh support nodes are very close to each other, thus generating many additional elements, resulting in occlusion of uncertainties or of the underlying surface. To minimize this effect, blending is introduced and can be enabled for any of the geometric methods, with the goal of achieving opacity in cases where occlusion becomes a serious problem.

As an alternative to a scattered representation of the accuracy, mesh based error representations are introduced - uncertainty boundary and uncertainty volume, as presented in Figure 3.11. Not surprisingly, these methods rely strongly on blending in order to avoid occlusion. At the same time, they manage to transmit a more compact representation of the error levels of a surface, encouraging the development of intuitive interpretations - a solid shape delimited in space has more correspondences in the real world than a set of bars or points.

Nevertheless, in certain cases even these visualization techniques reach their limits. A problem noticed during this project refers to the fact that when two meshes are closely positioned, the probability space or uncertainty volumes might intersect. In this case, the geometry based methods deliver poor results, due to the incapacity of finding the corresponding surface and occlusion.

As mentioned in Chapter 2.4 and, more specifically, in Figure 2.16, we expect to encounter at least four types of measurable uncertainties: uncertainty added by the noise in the TOF sensor data, uncertainty in the computed local patches, uncertainty through repeated point cloud matching and merging and uncertainty introduced by mesh multi-resolution methods. All these error values can be represented by using the representations from Figure 3.10 and Figure 3.11.

The current thesis implements only the computation and visualization of the filtering and multi-resolution error types. Still, other sources of uncertainty can be easily employed by using the same representation. Additionally, by using multiple proposed uncertainty visualization methods, we can differentiate between the various

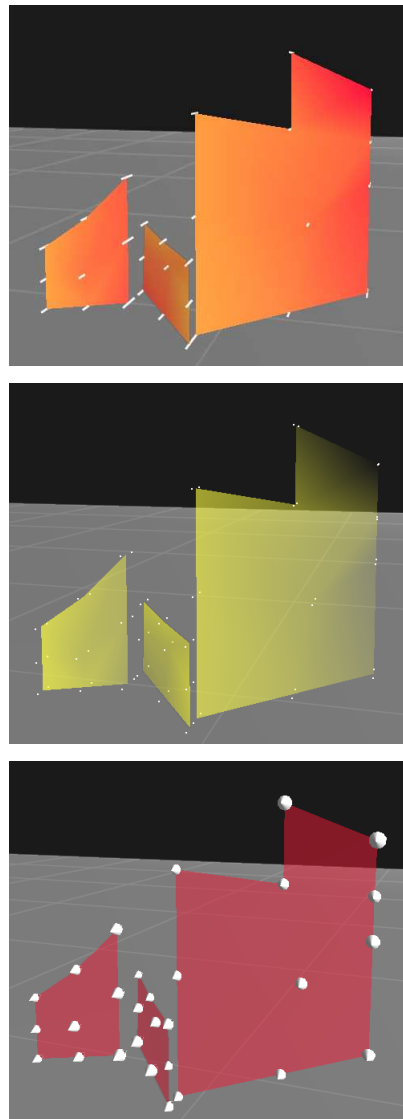


Figure 3.12: Separate visualization methods for different surface uncertainties by simultaneous use of multiple techniques - color mapping for filtering uncertainty and bars for multi-resolution uncertainty (top), transparency for filtering uncertainty and points for multi-resolution uncertainty (middle), cone with for filtering uncertainty and cone height for multi-resolution uncertainty (bottom).

errors introduced in the reconstruction process and their importance and influence. This approach is highlighted in Figure 3.12, where the noise reduction and multi-resolution uncertainties are displayed cumulatively. Also, Figure 3.12c presents a previously discussed method for adding geometrical glyphs that have the advantage of incorporating multiple levels of uncertainty in a single geometrical shape, in this case width and height.

A different approach considers the usage of the same uncertainty visualization method for multiple levels of uncertainty (Figure 3.13). Notice that the possibility of employing the same representation for different errors and still be able to distinguish between them is mostly available for the geometric methods. While it is sometimes possible to have multiple uncertainties and represent them all with colors and transparency, it has the distinct drawback of limiting the differentiation process. Nonetheless, approaches like the bar or point based visualization manage to minimize the occlusion and cluttering, especially when the surface features are down-sampled. but most of all, the uncertainty volumes manage to incorporate the highest amount of information and maintain it visible from almost any angle. A particular advantage of the geometric methods is that they are not limited, allowing for the parallel representation of any number of uncertainty levels.

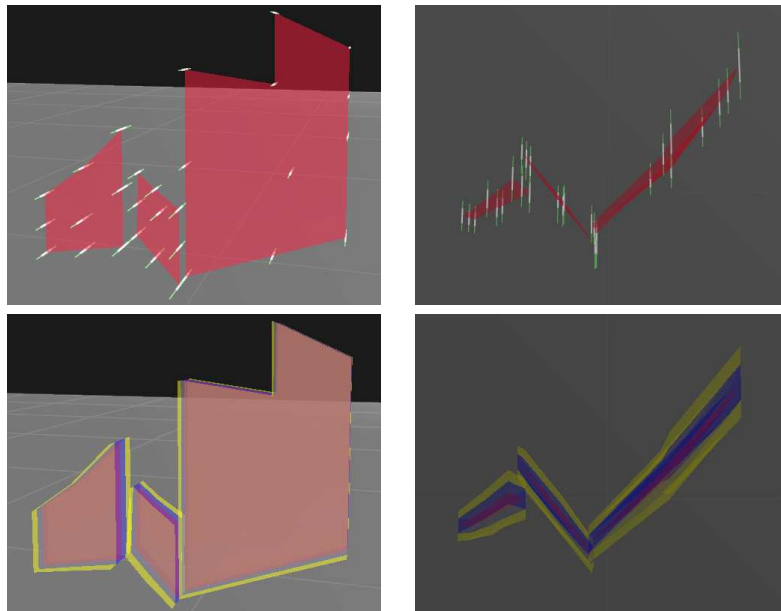


Figure 3.13: Identical visualization methods for different surface uncertainties by simultaneous use of multiple techniques - cumulative uncertainty representation via bars (left) and volume (right).

In cases with many sources of uncertainty and thus many additional geometric structures, the possibility for selective representation of the information is desired. As such, the different error levels can be enabled and disabled, or rearranged inside the representational pipeline, to allow for maximal flexibility and minimal information clutter.

On one hand, the use of additive uncertainty representation enables the estimation of the overall possible error in the computation of the optimal surface, as this can be



important in various tasks - a robot trying to navigate might be more interested in the closest possible surface, even if it has much lower probability than the optimal one. On the other hand, knowing exactly what error source influences the results most negatively and being able to create a more complete environment reconstruction to be submitted for user analysis, allows the better distribution of a limited resource: *time*.

When working with a real-time application, like the interpretation of 3D point clouds and the reconstruction of the sensor's surroundings from them, setting priorities is vital. As such, computing and visualizing multiple uncertainty levels allows for the objective selection of the most distortive one, towards devising / implementing a likely more computationally complex algorithm to limit its effects. In other words, one can minimize the error that introduces the maximal uncertainty.

In terms of performance, the uncertainty visualization methods have a limited influence on the rendering speed of each frame. Even if visualizing the error levels can double or triple the number of displayed vertices (i.e. the surface and volume uncertainty representations create additionally one or two maximal uncertainty meshes), they also are correlated with the current displayed LOD of the underlying estimated surface. Therefore, the rendering time can be reduced when additional noise levels are displayed by decreasing the LOD of the main surfaces and corresponding uncertainties. Furthermore, our tests have shown that a sufficiently high frame rate (over 25 fps) can be maintained when visualizing tens of meshes and uncertainties in a complex representation (i.e. lighting, blending etc.).



## Chapter 4

---

# Conclusions and Discussion

---

Throughout this thesis we propose a new model for virtual environment reconstruction from successive 3D range point cloud scans, with an emphasis on high precision and good performance, obtained by algorithms mostly used in the fields of graphics and visualization.

The problem of converting a series of point clouds to a surface-based environment is decomposed into multiple stages, as presented in Figure 3.1. A great advantage of the processing model is that the execution time of all the stages is small, allowing for the 3D mesh computation and rendering to be performed online.

In the beginning of the pipeline, instead of using algorithms that strive to be noise insensitive, we introduce a preprocessing step in order to minimize the presence of noise in the data and to supply enhanced distance images to the actual processing algorithms. To achieve the desired surface error reduction without eliminating vital edge information, we use anisotropic diffusion and wavelet denoising with main phase preservation. The parameters of these filters are tuned for maximal surface noise removal and optimal boundary preservation between surfaces scanned by a SwissRanger sensor that generates a 2D grid aligned 3D point clouds. By using this approach, the normalized range image is altered such that a high percentage of the scans can be successfully denoised. Additionally, we introduce an inexpensive and efficient denoising algorithm that not only eliminates false positives in Canny edge detection, but also allows for reconstruction of undetected boundary information in the original data.

For the actual reconstruction and rendering of the surfaces, we implement a multi-resolution method that is executed on every mesh surface independently. The idea here is to counteract the constant complexity increase of the overall representation with every set of surfaces and corresponding new vertices that are added. Besides the

fact that the LOD is reduced linearly with the increasing number of mesh nodes, the complexity of the surfaces is also guided by the distance to the viewer - the further a mesh is from the viewport, the less vertices it uses to be represented.

Note that the surface complexity is reduced by eliminating the least important nodes, one by one, by dynamically computing the sharp feature value of the surface at each point. During this process, the approximate boundary shape of each mesh is maintained, thus enabling a minimal information loss for quasi-planar surface representation. Additionally, a mesh merging approach is devised, which considers the previous multi-resolution scheme and reimplements it for the resulting mesh.

Finally, we have explored multiple surface uncertainty visualization methods that allow for estimating the environment more completely and devising optimal positioning and tracking algorithms by considering the represented maximal error levels. Furthermore, various error sources that affect the surface reconstruction process were visualized separately, as well as convoluted, offering support for tasks as targeted error handling/reduction and in-depth data analysis.

All in all, this paper supplies a new approach for the virtual reconstruction of surroundings from 3D point cloud scans, by exploiting the advantages of both graphics and robotics techniques, in order to achieve an efficient and precise environment reconstruction for robot understanding and navigation, as well as implement a user-intuitive representation for human interpretation.

## Chapter 5

---

# Future Work

---

Considering the overall complexity of the devised point cloud processing model (Figure 3.1), but also the intricate nature of each of the composite algorithms, the development directions that can be followed are numerous. Additionally, during this research several questions remained yet to be answered, mostly due to temporal constraints.

From a broad angle, we have to further elaborate and refine the entire process of converting a sequence of 3D point cloud scans into surfaces, all of this in real-time. In order to achieve this, various modules of the model have to be modified or yet implemented.

An important aspect for coherence and online functionality is the reimplementation of the noise reduction module in C++, to allow a real-time interaction with the following processing stages. Moreover, this would certainly boost the execution times for the filtering and enable us to consistently compute the computational expenses of each stage.

Of high importance is also the evaluation of other error reduction methods, especially denoising techniques that consider phase preservation, as these have already proven to be successful. To obtain a continuous processing, a more efficient surface registration and segmentation method has to be implemented, in order to eliminate the necessity for any human intervention. This can be achieved by use of preexisting methods, as surface registration and matching are not direct topics of this thesis.

In the presented approach, the multi-resolution methods are influenced only by the size of the global reconstruction - number of vertices in meshes - and the distance from each mesh to the screen. To improve this aspect, we consider implementing an extension that additionally manages occluded meshes and changes the LOD distance

scheme to redesign the previous heuristic one, by setting the detail level such that no two vertices of a mesh will be displayed in the same pixel. Also, in order to preserve sharp features for complex meshes, different vertex evaluation methods have to be considered.

An increase in speed can be also achieved at this stage by implementing the occlusion detection and mesh merging by means of the OpenGL depth buffer, that has the added bonus of simplifying the implementation.

Another direction that should be further pursued is the uncertainty visualization. Experimenting with other visual representations for uncertainty, evaluating their advantages and disadvantages from human and algorithmical view-point, handling overlapping uncertainty volumes, further exploring the possibilities of multi-level uncertainty representation and its limitations by considering additional information, like odometry, are only a couple of areas that would required attention.

Finally, probably the most important and challenging unsolved task is represented by the extension of the procedural model to incorporate and correctly solve the problem of reconstructing highly non-planar surfaces.

---

# Acknowledgements

---

I would like to express my feelings of gratitude to my supervisors, Prof. Lars Linsen and Prof. Andreas Birk, for their limitless encouragement, stimulating suggestions and positive reinforcement throughout the research, implementation and writing of this thesis.

Many thanks to my colleagues from the Visualization and Computer Graphics Laboratory and the Robotics Group at Jacobs University Bremen, for the inspiring discussions and sparked ideas, which enabled for a broad view and an in-depth exploration of the suggested topics.

My special thanks goes to my cat, Miuk, for scratching me when I was loosing momentum or concentration, but also for the fun moments we had when I needed to take a break.

Last but not least, I would like to thank my amazing family and dearest friends, who through their love and encouragement gave me the power to push on and without whom this thesis would never have come into existence.





---

# Bibliography

---

- [1] Peter Alfeld. *Scattered Data Interpolation in Three or More Variables*, volume Mathematical Methods in Computer Aided Geometric Design, Pages 1-33. Academic Press, 1989. [cited at p. 66]
- [2] Gerhard Heinrich Bendels, Patrick Degener, Roland Wahl, Marcel Krtgen, and Reinhard Klein. *Image-Based Registration of 3D-Range Data using Feature Surface Elements*. International Symposium of Virtual Reality, Archaeology and Cultural Heritage (VAST 2004), 2004. [cited at p. 26]
- [3] Paul J. Besl and Neil D. McKay. *A Method for Registration of 3-D Shapes*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 2, Pages 239-256, February 1992. [cited at p. 26, 70]
- [4] Stephan Bischoff and Leif Kobbelt. *Teaching Meshes, Subdivision and Multiresolution Techniques*. Computer-Aided Design 36(14), Pages 1483-1500, 2004. [cited at p. 35, 38]
- [5] Michael J. Black, Guillermo Sapiro, David H. Marimont, and David Heeger. *Robust Anisotropic Diffusion*. IEEE Transactions on Image Processing, Volume 7, No. 3, March 1998. [cited at p. 19, 20]
- [6] OpenGL Architecture Review Board, Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2 (5th Edition)*. Addison-Wesley Professional, August 2005. [cited at p. 50]
- [7] Dorit Borrmann, Jan Elseberg, Kai Lingemann, Andreas Nüchter, and Joachim Hertzberg. *The Efficient Extension of Globally Consistent Scan Matching to 6 DoF*. Technical Report, GT-IC-08-05 from the Georgia Institute of Technology, Atlanta (GA), USA, Pages 29-36, Proceedings of the 4th International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT '08), June 2008. Dataset available at <http://kos.informatik.uni-osnabrueck.de/3Dscans>, Section 6. [cited at p. 9, 49]
- [8] V. Caselles, G.Haro, G. Sapiro, and J.Verdera. *On Geometric Variational Models for Inpainting Surface Holes*. Computer Vision and Image Understanding, Vol. 111, Issue 3, Pages 351-373, September 2008. [cited at p. 32]
- [9] E. Catmull and J. Clark. *Recursively generated B-spline surfaces on arbitrary topological meshes*. Computer Aided Design 10, Pages 350-355, October 1978. [cited at p. 37]

- [10] Abdallah K. Cherri and Mohammad A. Karim. *Optical Symbolic Substitution: Edge Detection using Prewitt, Sobel, and Roberts Operators*. Applied Optics, Vol. 28, Issue 21, Pages 4644-4648, 1989. [cited at p. 22]
- [11] Jonathan Cohen, Marc Olano, and Dinesh Manocha. *Appearance-Preserving Simplification*. International Conference on Computer Graphics and Interactive Techniques, Proceedings of the 25th annual conference on Computer Graphics and Interactive Techniques, Pages 115-122, 1998. ACM, New York, NY, USA. [cited at p. 33]
- [12] CSEM. *The SwissRanger*, volume Manual v1.02. CSEM SA, Zurich 8048, Switzerland, 2006. <http://www.swissranger.ch>. [cited at p. 8, 23]
- [13] Brian Curless. *New Methods for Surface Reconstruction from Range Images*. Ph.D. dissertation, Technical Report CSL-TR-97-733, Stanford University, June 1997. [cited at p. 25, 30, 40]
- [14] Brian Curless and Marc Levoy. *A Volumetric Method for Building Complex Models from Range Images*. International Conference on Computer Graphics and Interactive Techniques, Pages 303-312, 1996. [cited at p. 12, 25, 27, 32, 39]
- [15] Olivier Devillers. *Improved Incremental Randomized Delaunay Triangulation*. Proceedings of the 14th Annual Symposium on Computation Geometry, Pages 106-115, June 1998. [cited at p. 63]
- [16] D. Doo. *A subdivision algorithm for smoothing down irregularly shaped polyhedrons*. Interactive Techniques in Computer Aided Design, Pages 157-165, Bologna, Italy, IEEE Computer Society, 1978. [cited at p. 37]
- [17] D. Doo and M. Sabin. *Behaviour of recursive division surfaces near extraordinary points*. Computer-Aided Design 10, Pages 356-360, September 1978. [cited at p. 37]
- [18] Tsung-Pao Fang and Les A. Piegl. *Delaunay Triangulation Using a Uniform Grid*. IEEE Computer Graphics and Applications, Volume 13, Issue 3, Pages 36-47, May 1993. [cited at p. 63]
- [19] D. Fischer and P. Kohlhepp. *3D Geometry Reconstruction from Multiple Segmented Surface Descriptions using Neuro-Fuzzy Similarity Measures*. Journal of Intelligent and Robotic Systems, Vol. 29, Pages 389-431, 2000. [cited at p. 27]
- [20] Thomas Gerstner and Renato Pajarola. *Topology Preserving and Controlled Topology Simplifying Multiresolution Isosurface Extraction*. IEEE Visualization Conference (VIS 2000), 2000. IEEE Computer Society. [cited at p. 37]
- [21] Craig Gotsman, Stefan Gumhold, and Leif Kobbelt. *Simplification and Compression of 3D Meshes*. [cited at p. 33]
- [22] Craig Gotsman, Stefan Gumhold, and Leif Kobbelt. *Surface Simplification using Quadric Error Metrics*. International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), Pages 209-216, 1997. [cited at p. 35]
- [23] Gevorg Grigoryan and Penny Rheingans. *Point-Based Probabilistic Surfaces to Show Surface Uncertainty*. IEEE Transactions on Visualization and Computer Graphics, Volume 10, No. 5, October 2004. [cited at p. 42, 43, 47]

- [24] A. Nuechter, H. Surmann, and J. Hertzberg. *An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments*. Robotics and Autonomous Systems, Vol. 45, No. 3-4, Pages 181-198, 2003. [cited at p. 8, 23]
- [25] J. A. Hartigan and M. A. Wong. *A K-Means Clustering Algorithm*. Applied Statistics 28, Pages 100-108, 1979. [cited at p. 15, 16]
- [26] Dirk Hähnel, Wolfram Burgard, and Sebastian Thrun. *Learning Compact 3D Models of Indoor and Outdoor Environments with a Mobile Robot*. Robotics and Autonomous Systems, Volume 44, Pages 15-27, 2003. [cited at p. 34]
- [27] Hugues Hoppe. *Progressive Meshes*. ACM SIGGRAPH Conference Proceedings, Pages 99-108, 1996. [cited at p. 35]
- [28] Hugues Hoppe. *View-Dependent Refinement of Progressive Meshes*. ACM SIGGRAPH Conference Proceedings, Pages 189-198, 1997. [cited at p. 35]
- [29] Hugues Hoppe, Tony DeRose, Tom Duchampy, John McDonald, and Werner Stuetzle. *Surface Reconstruction from Unorganized Points*. International Conference on Computer Graphics and Interactive Techniques, Pages 71-78, 1992. [cited at p. 13]
- [30] Khronos Group Inc. *OpenGL: The Industry Foundation for High Performance Graphics*. Online. <http://www.opengl.org>. [cited at p. 49]
- [31] Christopher Johnson and Charles Hansen. *Handbook of Visualization*. 2004. Academic Press, Inc., Orlando, FL, USA. [cited at p. 93, 94]
- [32] Wen-Chung Kao, Ying-Ju Chen, Chia-Ping Shen, Chi-Wu Huang, and Sheng-Yuan Lin. *Integrating Edge Detector and Bilateral Noise Filter for Enhancing Color Images*. IEEE International Symposium on Circuits and Systems, 2006. [cited at p. 22]
- [33] Andreas Kerren, Achim Ebert, and Jörg Meyer. *Human-Centered Visualization Environments*. Image Processing, Computer Vision, Pattern Recognition, and Graphics, Vol. 4417, 2007. [cited at p. 44, 46]
- [34] Rolf Klein. *Algorithmische Geometrie*. Springer, Page 304, 2005. [cited at p. 63]
- [35] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. *Interactive Multi-Resolution Modeling on Arbitrary Meshes*. International Conference on Computer Graphics and Interactive Techniques, Proceedings of the 25th annual conference on Computer Graphics and Interactive Techniques, Pages 105-114, 1998. ACM, New York, NY, USA. [cited at p. 37, 38]
- [36] Peter Kovsi. *Phase Preserving Denoising of Images*. DICTA'99, The Australian Pattern Recognition Society Conference, Perth WA, Pages 212-217, December 1999. [cited at p. 17, 56]
- [37] Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. *MAPS: Multiresolution Adaptive Parameterization of Surfaces*. Computer Graphics Journal, Volume 32, Pages 95-104, 1998. [citeseer.ist.psu.edu/article/lee98maps.html](http://citeseer.ist.psu.edu/article/lee98maps.html). [cited at p. 35, 36, 37]
- [38] Lars Linsen. *Point Cloud Representation*. Technical Report, Fakultät für Informatik, Universität Karlsruhe, 2001. [cited at p. 12]

- [39] HongGen Luo, LiMin Zhu, and Han Ding. *Coupled Anisotropic Diffusion for Image Selective Smoothing*. Signal Processing, Volume 86, Issue 7, Pages 1728-1736, 2006. Elsevier North-Holland Inc., www.ElsevierComputerScience.com. [cited at p. 19]
- [40] Niloy J. Mitra and An Nguyen. *Estimating Surface Normals in Noisy Point Cloud Data*. International Journal of Computational Geometry and Applications, Volume 14, Pages 261-276, 2004. [cited at p. 13]
- [41] Nikhil R. Pal and Sankar K. Pal. *A Review on Image Segmentation Techniques*. Pattern Recognition, Vol. 26, No. 9, Pages 1277-1294, 1993. [cited at p. 25]
- [42] Alex T. Pang, Craig M. Wittenbrink, and Suresh K. Lodha. *Approaches to Uncertainty Visualization*. The Visual Computer, Volume 13, Pages 370-390, 1997. [cited at p. 39, 42]
- [43] Sung W. Park, Lars Linsen, Oliver Kreylos, and John D. Owens. *Discrete Sibson Interpolation*. IEEE Transactions on Visualization and Computer Graphics, Volume 12, Issue 2, Pages 243-253, March 2006. [cited at p. 67]
- [44] Pietro Perona and Jitendra Malik. *Scale-Space and Edge Detection using Anisotropic Diffusion*. IEEE Transactions on Pattern Analysis and Machine Intelligence 12, Pages 629-639, 1990. [cited at p. 18, 19, 53]
- [45] Jörg Peters. *Patching Catmull-Clark meshes*. Siggraph 2000, Pages 255-258, 2000. Addison Wesley Longman. [cited at p. 25, 37]
- [46] Jann Poppinga, Narunas Vaskevicius, Andreas Birk, and Kaustubh Pathak. *Fast Plane Detection and Polygonalization in Noisy 3D Range Images*. 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Acropolis Convention Center, Nice, France, September 2008. [cited at p. 9, 23, 25, 34]
- [47] Szymon Rusinkiewicz and Marc Levoy. *Efficient Variants of the ICP Algorithm*. Third International Conference on 3-D Digital Imaging and Modeling, Pages 145-152, Quebec City, Quebec, Canada, 2001. [cited at p. 26]
- [48] G. Salemi, F. Liberi, S. Mischi, V. Achilli, and G. Artese. *Multi-resolution Modelling from Multiple Range Views: The Laser Scanner Survey of Porta Portello, Padua, Italy*. International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences, Vol. 35, Part 5, Pages 242-245, 2004. [cited at p. 37]
- [49] Oliver Schall, Alexander Belyaev, and Hans-Peter Seidel. *Robust Filtering of Noisy Scattered Point Data*. Eurographics Symposium on Point-Based Graphics (2005), June 2005. [cited at p. 16]
- [50] Hanno Scharr, Michael J. Black, and Horst W. Haussecker. *Image Statistics and Anisotropic Diffusion*. ICCV, Proceedings of the Ninth IEEE International Conference on Computer Vision, Volume 2, Pages 840-847, 2003. IEEE Computer Society. [cited at p. 19, 20]
- [51] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Upper Saddle River, NJ: Prentice Hall, 2001. Section 5. Filtering and Enhancing Images. [cited at p. 16, 21]
- [52] Jochen Süßmuth and Günther Greiner. *Ridge Based Curve and Surface Reconstruction*. Proceedings of the 5th Eurographics Symposium on Geometry Processing, Pages 243-251, 2007. [cited at p. 25]

- [53] Marc Soucy and Denis Laurendeau. *Multi-resolution Surface Modeling from Multiple Range Views*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Pages 348-353, June 1992. [cited at p. 29, 30, 31]
- [54] Hartmut Surmann, Andreas Nuchter, and Joachim Hertzberg. *An Autonomous Mobile Robot with a 3D Laser Range Finder for 3D Exploration and Digitalization of Indoor Environments*. Robotics and Autonomous Systems 45, Pages 181-198, Fraunhofer Institute for Autonomous Intelligent Systems (AIS), Schloss Birlinghoven, D-53754 Sankt Augustin, Germany, September 2003. [www.ComputerScienceWeb.com](http://www.ComputerScienceWeb.com). [cited at p. 38]
- [55] Juan D. Tardòs. *Representing Partial and Uncertain Sensorial Information Using the Theory of Symmetries*. IEEE International Conference on Robotics and Automation, Volume 2, Pages 1799-1804, May 1992. [cited at p. 41, 44]
- [56] Tolga Tasdizen, Ross Whitaker, Paul Burchard, and Stanley Osher. *Geometric Surface Smoothing via Anisotropic Diffusion of Normals*. Proceedings of the conference on Visualization '02, Pages 125-132, Boston, Massachusetts, 2002. [cited at p. 20, 34]
- [57] Greg Turk and Marc Levoy. *Zippered Polygon Meshes from Range Images*. International Conference on Computer Graphics and Interactive Techniques, Pages 311-318, 1994. [cited at p. 28, 30]
- [58] Narunas Vaskevicius, Andreas Birk, Kaustubh Pathak, and Jann Poppinga. *Fast Detection of Polygons in 3D Point Clouds from Noise-Prone Range Sensors*. International Workshop on Safety, Security and Rescue Robotics, Rome, Italy, September 2007. [cited at p. 23, 25, 34]
- [59] Craig M. Wittenbrink, Alex T. Pang, and Suresh Lodha. *Verity Visualization: Visual Mappings*. Technical Report UCSC-CRL-95-48, University of California at Santa Cruz, Santa Cruz, CA, USA, May 1995. [cited at p. 41]
- [60] Richard S. Wright, Benjamin Lipchak, and Nicholas Haemel. *OpenGL SuperBible: Comprehensive Tutorial and Reference (4th Edition)*. Addison-Wesley Professional, June 2007. [cited at p. 50]
- [61] Yongxin Yu and Zhiyong Feng. *An Image Segmentation Method Based on Adaptive Anisotropic Diffusion Filtering*. Proceedings of the Second International Multi-Symposiums on Computer and Computational Sciences, Pages 177-181, 2007. IEEE Computer Society. [cited at p. 19, 23, 25, 34]
- [62] Hongyuan Zha, Xiaofeng He, Chris H. Q. Ding, Ming Gu, and Horst D. Simon. *Spectral Relaxation for K-means Clustering*. NIPS, Pages 1057-1064, 2001. [cited at p. 13, 15]
- [63] Zhengyou Zhang. *Iterative Point Matching for Registration of Free-form Curves*. International Journal of Computer Vision, Vol. 13, No. 3, Pages 119-152, October 1994. [cited at p. 26, 70]



# Appendices





## Appendix A

---

# Terminology

---

- Point Cloud - digitalized data containing a set of 3D points (each point presenting at least 3 values for the 3 coordinates -  $X$ ,  $Y$ ,  $Z$ ) defining part of an object or environment. The data is usually generated with some measurement device (i.e. time-of-flight sensor) and it represents the object's boundary, but in rare cases it can be generated computationally.
- Surface - an infinite densely connected manifold, referring usually to the 2D and 3D cases. A surface can be positioned and oriented in space, and it can either have a boundary or extend infinitely. A planar surface is a particular case, where all the points of the surface are included in a 2D manifold and the normal at each point is the same. Mathematically, a surface can be represented either implicitly ( $F(x, y, z) = 0$ ) or explicitly ( $F(x, y, z) = a$ ).
- Rendering - the process of generating a drawing/image on a graphical output device from a model, by means of computer programs. The model is a strict description in a particular language or code of the 3D objects. In general, rendering a scene would suppose the presence of geometrical data, viewpoint, texture data, lighting and shading information. Common rendering techniques include ray-tracing and scanline rendering. Additionally, rendering can be divided in geometry-based rendering or image-based rendering [31].
- Level of detail (LOD) - the amount of detail or complexity of an object or environment that is computed in order to be displayed at a moment in time. The higher the LOD, the more time it will take for the object to be rendered on a computer screen and the more probable it is that the displayed object will be more accurate/complex, and vice versa. LOD is usually correlated in graphics with the distance between the rendered object in the virtual 3D space and the projection surface (i.e. the screen).

- Pixel - the smallest piece of information in a digital image (**P**icture **E**lement). Pixels are displayed in a regular 2D grid and are in general represented by small squares or dots. Each pixel is a sample at a certain position of the original image. Thus, more pixels lead to a denser sampling, which again leads to a more accurate representation of the original image.
- Mesh - a set of planar faces joined together along their edges. As such, these faces do not overlap and they can present 3 or more edges. The most common mesh is the *triangular mesh*, where the planar faces are 2D triangles. The structural elements of a mesh are the faces, the edges and the vertices.
- Modeling - the generation of a computer implemented object defined by a set of 3D points. By uniting pairs of such points by lines (e.g. tessellation), one can obtain structures representing multi-dimensional objects. Such a simple representation that is based on lines only is called a *wireframe*.
- Simultaneous localization and mapping (SLAM) - a technique designed for mobile robots with the purpose of building a map within an unknown environment, and at the same time keeping track of their current position. One major difficulty in this process is represented by the inherent uncertainties in discerning the robot's relative movement from the provided sensors, as using only the odometry sensors proves to be generally inefficient.
- OpenGL - a standard specification developed by Silicon Graphics Inc. (SGI) in 1992, defining a cross-language cross-platform programming interface for creating rich 2D and 3D computer graphics applications. The interface enables the use of multiple graphical primitives that allow the composition of complex three-dimensional (3D) scenes. Areas that use OpenGL intensively include virtual reality, scientific visualization, information visualization, flight simulation, as well as video games and 3D animation, where it competes with Direct3D on Microsoft Windows platforms.

For further information on terminology in this proposal please refer to [31].

---

# List of Symbols and Abbreviations

---

Abbreviation	Description	Definition
FPS	Frames Per Second	page 9
ICP	Iterative Closest Point	page 26
LOD	Level Of Detail	page 93
LRF	Laser Range Finder	page 8
PM	Progressive Meshes	page 35
RBF	Radial Basis Function	page 32
RMS	Root Mean Square	page 17
SLAM	Simultaneous Localization And Mapping	page 9
SR	Swiss-ranger	page 8
TOF	Time Of Flight	page 7

---

# List of Figures

---

1.1	3D point cloud rendering example from a music video . . . . .	3
1.2	Grayscale normalized distance image from TOF sensor . . . . .	4
2.1	Multiple range images of the same object from different angles . . . . .	8
2.2	Two Rugbots (rugged robot) in a rescue scenario . . . . .	9
2.3	$N \times D$ spacial representation of range point clouds . . . . .	15
2.4	Intensity discontinuity and random pixel neighborhood . . . . .	19
2.5	Noise filters on a point cloud with 6 surfaces . . . . .	21
2.6	Noise filters on a point cloud with 9 surfaces . . . . .	22
2.7	Laplacian edge detection results on range point clouds . . . . .	23
2.8	Two point clouds and correspondences computed with ICP . . . . .	26
2.9	Merging of two partially overlapping meshes . . . . .	28
2.10	Surface manifold representing the extended boundary of the mesh . . . . .	29
2.11	Boundary interpolation between two non-overlapping meshes . . . . .	30
2.12	Processing steps from point cloud to surface mesh . . . . .	31
2.13	Smoothed mesh model obtained from 3D point cloud . . . . .	35
2.14	Fine and coarse resolution mesh representation . . . . .	36
2.15	Uncertainty pipeline . . . . .	39
2.16	The 4 levels of uncertainty in our mesh visualization . . . . .	41
2.17	Point-based surface uncertainty visualization . . . . .	42
2.18	Surface uncertainty visualization by modifying geometry or attributes . . . . .	44
2.19	Glyph representation of population density . . . . .	45
2.20	Variable visual attributes for geometrical objects . . . . .	46
2.21	Hue-based surface uncertainty visualization . . . . .	46
3.1	3D point cloud processing model . . . . .	50
3.2	Scene photography, normalized distance image and edge detection . . . . .	52
3.3	Edge detection algorithms on original and filtered point clouds . . . . .	55
3.4	$N \times D$ representation of original and filtered point clouds . . . . .	60
3.5	Distance image transformation and mapping in $N \times D$ space . . . . .	61

3.6	Denoising and edge preservation method with 6 surfaces . . . . .	62
3.7	Denoising and edge preservation method with 9 surfaces . . . . .	63
3.8	Multi-resolution representation of a surface mesh via Shepard . . . . .	67
3.9	Merging two corresponding surfaces . . . . .	71
3.10	Uncertainty visualization by attribute addition . . . . .	72
3.11	Uncertainty visualization by geometry addition . . . . .	73
3.12	Separate methods for different surface uncertainties . . . . .	75
3.13	Identical methods for different surface uncertainties . . . . .	76



---

# Originality of the work

---

I, Daniel Cernea, hereby state that to the best of my knowledge this work contains original and independent results that had not been submitted elsewhere for conferral of a degree.