3D SURFACE RECONSTRUCTION IN TELEPRESENCE SYSTEMS

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Laura Mora Ballestar

In partial fulfilment

of the requirements for the degree in

Audiovisual Systems ENGINEERING

Advisor: Javier Ruiz Hidalgo

Barcelona, June 2017

ABSTRACT

Registering three-dimensional models is possible thanks to the use of multi view captures or the readings provided by depth sensors, which can be applied in Augmented or Virtual Reality systems. In this sense, this project focuses on a Telepresence system, which aims to register a room with depth sensors to be visualized by an external person, who is immersed in this new environment using Virtual Reality glasses. For this reason, the objective of the thesis is to integrate a 3D surface reconstruction method in the Telepresence system. For this, several algorithms have been studied, evaluating the quality of the reconstruction in different conditions, both in the application of the algorithms and in the data where the algorithms are applied. Next, the one that best suits the Telepresence platform is selected and integrated. The results show that few algorithms are suitable for our environment, but it is possible to integrate them in the platform.

RESUM

Enregistrar models tridimensionals és possible gràcies a l'ús de captures multi view o de lectures proporcionades per sensors de profunditat que poden ser aplicades en sistemes de Realitat Augmentada o Virtual. En aquest sentit, el projecte es centra en un sistema de Telepresència, que té per objectiu enregistrar una sala amb sensors de profunditat i que una persona externa es submergeixi en aquest nou entorn fent ús d'ulleres de Realitat Virtual. Per aquest motiu, l'objectiu de la tesis és integrar un mètode de reconstrucció de superfícies 3D en el sistema de Telepresència. Per fer-ho, s'ha estudiat diversos algoritmes, avaluant-ne la qualitat de la reconstrucció en diferents condicions, tant en l'aplicació dels algoritmes com en les dades sobre les que es treballa. Seguidament, es selecciona i s'integra el que millor s'adapta al nostre entorn de treball. Els resultats mostren que pocs algoritmes són aptes pel nostre entorn, però si n'és possible la integració a la plataforma.

RESUMEN

Registrar modelos tridimensionales es posible gracias al uso de capturas multi-view o las lecturas proporcionadas por sensores de profundidad que pueden ser aplicadas en sistemas de Realidad Aumentada o Virtual. En este sentido, el proyecto se centra en un sistema de Telepresencia, que tiene por objetivo registrar una sala con sensores de profundidad y que una persona externa se sumerja en este nuevo entorno haciendo uso de gafas de Realidad Virtual. Por este motivo, el objetivo de la tesis es integrar un método de reconstrucción de superficies 3D en el sistema de Telepresencia. Para ello, se ha estudiado varios algoritmos, evaluando la calidad de la reconstrucción en diferentes condiciones, tanto en la aplicación de los algoritmos como en los datos sobre los que se trabaja. Seguidamente, se selecciona el que mejor se adapta y se integra en nuestro entorno de trabajo. Los resultados muestran que pocos algoritmos son aptos para nuestro entorno, pero si es posible la integración en la plataforma.

“...o pots fer al que a ella li agradaria:

somriure, obrir els ulls, estimar i seguir”

ACKNOWLEDGEMENTS

I would like to thank my tutor Javier Ruiz for all the help and guidance provided during the project. I would also like to thank all the Telepresence group, Arantxa Casanova, Belén Luque and Eduard Pallàs for delivering such a nice platform and providing me with the data to develop my thesis. Moreover, I would like to specially thank Josep Ramon Casas for his advices and Albert Gil for all the assistance.

In addition, thank to Stanford Repository and Aim@Shape for providing opensource meshes.

Finally, I would also like to thank my father and sister for listening and giving me advice when I needed it. And Gerardo Garcia for supporting me and providing another point of view to approach the problems.

CONTENT

FUNDAMENTALS AND ABBREVIATIONS

FUNDAMENTALS

Point Cloud: Set of points that define the model of an object in a three-dimensional space which intend to represent the external surface of the object. Each of these points is defined by its position (x,y,z) and its colour (r,g,b).

Mesh: Also called Polygon Mesh, is a collection of vertices, edges and faces which define a shape. The faces usually consist on triangle forms. The resulting surface is mathematically called "unstructured grid or tessellation".

Rendering: 3D computer graphics process to apply textural and lightning information to the mesh.

ABBREVIATIONS

PLY: Polygon File Format. It is a file format for storing graphical objects, described as Point Clouds or Meshes. http://paulbourke.net/dataformats/ply/.

PTS: File format for describing collections of points in a 3D space with only XYZ components.

OFF: Object File Format. It provides the definition of the polygons describing a 3D object.

ROS: Robot Operating System

XML: Extensible Markup Language. Language for descripting data.

CHAPTER 1. INTRODUCTION

1.1 MOTIVATION

Nowadays, Virtual and Augmented reality are exponentially growing and new sectors for its applications are being explored. Nonetheless, it is common to confuse both systems because they both share the ability to alter our perception of the world but exploiting such ability in different ways.

Virtual Reality (VR) refers to the technology that simulates the user's physical presence in a new environment and its software and hardware allows the user to interact with it, by moving around the scene or moving/touching objects. Typically, this environment is artificially generated and visualized in special headsets, that consist of a stereoscopic head-mounted display, motion tracking sensors and some brands incorporate hand controllers. On the contrary, Augmented Reality (AR) alters the real-world environment by adding computer-generated elements, such as 3D virtual graphics, text, images or videos. These information is showed to the user with distinct types of screens or goggles which allows the user to see the natural environment, without altering the user's physical presence location.

Currently, research is exploring the idea of linking both technologies, and it is being called "*Real Virtuality*" [1]. One of the possibilities is to register a natural scene using depth sensors and visualize this real-world environment with VR devices and, also, offering the chance to insert virtual objects into the scene, like in AR.

The result from capturing a natural scene with depth sensors is a cloud of organized points, called point cloud. A point cloud is a set of points that define a model of an object in a three-dimensional space which intend to represent the external surface of the object. Each of these points is defined by its position (x,y,z) and its color (r,g,b). The problem with the point cloud representation of an object is that it does not have surfaces, so you can see through the object. The process to generate the surfaces from point clouds is called 3D Surface Reconstruction, which creates connectivity between points in form of polygons, building a mesh. Figure 1 shows this process.

a) *b)* c)

Figure 1:From Point Cloud to Mesh a) Stanford Bunny Point Cloud b) Stanford Bunny 3D Surface Reconstruction (Mesh) c) 3D Surface Reconstruction after rendering

Seeing that, the purpose of the project is to provide a 3D Surface Reconstruction from a recorded scene captured with depth sensors.

1.2 PROJECT BACKGROUND

The project will be carried out at the Image Department at the UPC (GPI), inside the Telepresence project. The objective of Telepresence is to generate a mixed VR and AR platform, by recording the Smart Room from the D5 building with RGBD sensors (Kinect 2) and then the scene would be transmitted and visualized, in real time, into a Virtual Reality device like HTC vive. The Smart Room was created in 2006 with the purpose of providing an intelligent room equipped with multiple cameras and microphones to study visual perception and acoustics [2].

The Telepresence platform structure is the following:

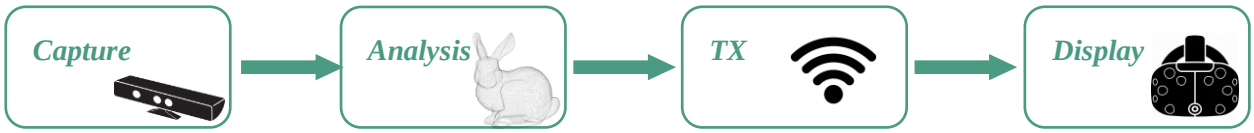


Figure 2: Telepresence Project Structure

The Telepresence platform is currently working and the idea of the Telepresence project is to integrate new features to the complete system, including the work that will be presented in this project, which will be integrated in the Analysis Block.

1.3 OBJECTIVES

To fulfill the purpose of the project, the main goals are:

1. Study the current 3D surface reconstruction algorithms to select the most appropriate for the project, considering the processing time versus the reconstruction quality.
2. Implement the chosen algorithm into the telepresence environment.
3. Optimize the algorithm to work in real-time, where the maximum delay between the captured data and its visualization is from the order of milliseconds.
4. Improve the 3D surface reconstruction algorithm to consider spatio-temporal consistency.

1.4 PROJECT STRUCTURE

The project has been developed for four months and we have divided the work accordingly to the objectives we wanted to fulfil. We have mainly divided the project into two parts: (1) the study and evaluation of the different 3D Surface Reconstruction algorithms and (2) the integration of a 3D Surface Reconstruction algorithm into the Telepresence platform. The possibility to implement the 3D Reconstruction in real-time will be studied during the integration part of the project.

The project breakdown structure has been divided in four Work Packages (WP) and the detailed tables and the Gantt diagram can be seen in Annex A.

1. **WP1 Analysis**: comprehends the first objective of the project.
2. **WP2 Implementation**: includes processing the Point Cloud and achieving the fourth objective.
3. **WP3 Integration**: involves the integration of a 3D surface reconstruction algorithm into the Telepresence platform, which encompasses the second and third goals.
4. **WP4 Documentation**: encloses all the reports for the project.

The organization of the present document will follow the breakdown structure explained before. Chapter 2 provides all the theoretical background foundations of the project. Chapter 3 and 4, detail the followed methodology and the obtained results for the WP1. Then, after the decision to which algorithm is the most suitable for the Telepresence platform is made, Chapter 5 explains the integration in the platform. Finally, Chapter 6 provides the budget of the project and Chapter 7 draws the main conclusions and future work.

CHAPTER 2. STATE OF THE ART

Since the 90's, there has been an augment of research involving 3D surface reconstruction. New depth sensors have appeared, and these have shown a wide new range of possibilities, such as applications for medicine, engineering and digital film-making. However, as said before, the data provided by these depth sensors is a point cloud that represents a natural scene, but it lacks on connectivity between the points.

By connecting the neighbors of the point cloud, it is possible to reconstruct the surface of the scene in form of polygons, such as triangles. This technique is called: polygonal mesh. This representation, with triangle polygons, is widely used in computer graphics due to its compactness and suitability for showing visual features. Many algorithms have appeared during several decades and a main classification, regarding how the surface is calculated has been done:

- **Reconstruction using Parametric Surfaces:** These algorithms use the existing points from the cloud to build the mesh. Examples of methods using parametric surface reconstruction are: Ball-Pivoting [3], VRIP [4] and Power Crust [5].
- **Reconstruction using Implicit Functions:** These algorithms compute an approximation of the input points to build the mesh. These methods are robust to density variation of the point cloud, but they need the surface normal for a point set. Examples of this type of reconstruction are: Marching-Cubes [6], Poisson [7], Screened Poisson [8], SSD [9], Fast Triangulation [10] and 3D Reconstruction [11].

Another possible classification of the algorithms is on how they need the data to be. This classification is independent of the reconstruction type explained in the first classification.

- **Reconstruction from unorganized points**: This data does not make assumptions about the point's connectivity. For this reason, they correctly behave in smooth surfaces but they commonly misbehave in complex surfaces, such as regions with high curvature or in point clouds with distortions or outliers.
- **Reconstruction from organized points:** This data uses an underlying structure of the acquired data, so the relationship between adjacent points is known. Generally, the reconstructions using organized points perform better than the unorganized. All data from depth sensors is organized.

One of the first 3D reconstruction algorithms to appear was Marching Cubes [6] and it has become a reference to more recent methods. The objective of this algorithm is to create triangle models of constant density for medical data. The algorithm scans the data with slices and uses an implicit function to estimate the triangle vertices by using linear interpolation. Once the tessellation is

completed, a gradient determines the direction of the original data to complete the render of the model.

Another reference method is the Ball-Pivoting [3] algorithm which is closely related to Alpha-Shapes [12]. This method uses a parametric framework to create the resulting mesh by using a Delaunay-Triangulation approach. For a set of points, the Delaunay-Triangulation ensures that the circumcircle associated with each triangle contains no other points inside. Using this idea, the Ball-Pivoting algorithm pivots a ball through the different edges of the triangles and builds the tessellation.

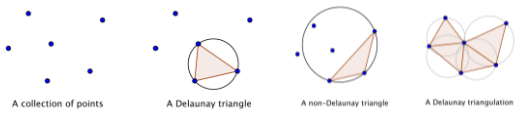


Figure 3: Delaunay Triangulation [https://mathmunch.org/2013/05/08/circling-squaring-and-triangulating/]

A newer method, but already well-stablished, is the Poisson Reconstruction Algorithm [7]. This method needs oriented points and it computes the reconstructed surface by approximating an indicator function, defined as 1 inside the model and 0 outside. Then, an appropriate isosurface (surface with constant value) is extracted to obtain the reconstructed surface. The algorithm considers all the points at the same time making the reconstruction very resilient to noisy data.

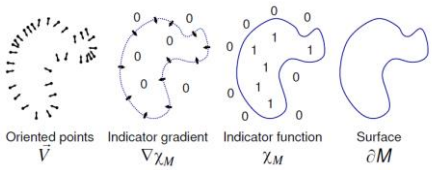


Figure 4: Poisson Reconstruction [5]. Example of the Poisson reconstruction process in 2D

A previous method from the Poisson Reconstruction authors was the Reconstruction of Solid Models from Oriented Point Sets (3D Reconstruction) [11], which computes the reconstruction following the same framework as Poisson. The difference relies on the approach for solving the indicator function. 3D Reconstruction uses a Fourier-based reconstruction whereas the Poisson system allows an adaptive discretization.

Some years later, the same authors defined the Screened Poisson Surface Reconstruction [8], that improved the Poisson Reconstruction by reducing the computation time and creating higher-quality surface reconstructions. The difference between both algorithms relies on the indicator function definition, which simplifies the derivations respect to the Poisson algorithm. Screened Poisson defines the indicator function as $\frac{1}{2}$ inside the model and $\frac{-1}{2}$ outside, so the isosurface passes near the input points of the cloud.

A similar algorithm is the SSD: Smooth Signed Distance Surface Reconstruction [9]. This method differs from the Poisson method by non-forcing the implicit function to approximate an indicator function, but forcing it to be a smooth approximation of the signed distance to the surface.

Another Surface Reconstruction method that solves the problem using the implicit function framework, but using unorganized points is the Fast Triangulation. This algorithm uses a greedy approach, where edges are directly written and never deleted. Last, one of the few methods using a Parametric reconstruction approach is Power Crust, which bases its reconstruction on the Voronoi Diagramas [13] and Boissanat [14]. The approach is to estimate the medial axis transform and inverse this transform to build the surface reconstruction.

In more recent years, new research has been directing towards applying meshes to moving objects and exploit the spatio-temporal consistency. One of the newer methods is from Microsoft Research, Fusion 4D [15], where they exploit the spatio-temporal consistency in challenging scenes, captured in a Multiview scenario with depth sensors, accomplishing the reconstructions in real time. Other methods researching in the temporal consistency are: Motion Graphs for Unstructured Textured Meshes [16], where they reconstruct the textured mesh at each frame, 4D Model Flow [17], which computes the appearance motion to build the 4D models and Hybrid Skeletal-Surface Motion Graphs [18] which uses a skeletal control of the models. From these methods exploiting the spatio-temporal consistency, Fusion 4D is the one providing better results, as it does not only work for people but it can perfectly reconstruct animals and objects with different mobility. It also reconstructs scenes with elevate level of details and movements. Compared with the other three algorithms, Motion Graphs for Unstructured Textured Meshes is the only one presenting results in difficult scenes, with dancers and rigid objects.

Other techniques that are currently being explored to improve the 3D Surface Reconstructions are post-processing techniques. An example of a post-processing is the work presented in [19] where a gradient-domain processing is applied to triangular meshes in 3D. With this technique, the meshes can be smoothed or sharpened.

CHAPTER 3. PROJECT METHODOLOGY

In this chapter, the methodology followed to develop the project will be explained. Figure 5, shows a workflow of the process to select and integrate a 3D Surface Reconstruction algorithm into the Telepresence platform.

Figure 5: Workflow of the Project

First of all, an extensive research of the well stablished 3D Surface Reconstruction algorithms as well as the most recent ones will be done. Then, some of these algorithms will be selected to be tested and analyzed. To examine the different methods, we will need to create an enough large database to have as much data as possible to perform a precise evaluation. To achieve an accurate evaluation, we will need to analyze the data with reliable measurements for comparing the quality of 3D Surface Reconstructions. Finally, we will need to compare the processing time with the reconstruction quality to reach a compromise between both measures, as the Telepresence system should work in real time. The last part of the workflow is the Integration block, which will be explained in CHAPTER 5.

3.1 SOFTWARE

The software used for the first part of the project is the following:

- **MeshLab** [19]: is an Open software mesh processing system that has been developed at the Visual Computing Lab of the ISTI-CN. MeshLab provides many functionalities, such as a large variety of input/output formats, mesh cleaning filters, remeshing filters, etc…

- **PCL** [20]: Point Cloud Library is a C++ library for processing 3D point clouds. It includes features such as filtering, feature estimation, surface reconstruction, model fitting, segmentation, etc …

3.2 ALGORITHM SELECTION

After performing an extensive research on the State of the Art 3D Surface Reconstruction algorithms, the methods that will be used for the evaluation are: 3D Reconstruction [11], Power Crust [5], Marching Cubes [6], Fast Triangulation [10], Poisson [7], Screened Poisson [8] and Smooth Signed Distance Surface Reconstruction (SSD) [9]. These methods vary from well-known algorithms such as Marching Cubes and Poisson to more recent ones like Screened Poisson and SSD.

These methods have been selected for several reasons. First, Marching Cubes and Poisson were selected for its history, as Marching Cubes is one of the first 3D Surface Reconstruction algorithms to appear and Poisson is a widely used method. Then, the 3D Reconstruction algorithm was selected as the results presented in [11] showed that the reconstruction of 10,000 points could be accomplished in less than 0.2 s, which is fastest than most algorithms. The Fast Triangulation algorithm was also chosen for the computing time showed in [10], where the surface from 20,000 points was reconstructed in 0.8 s. The Power Crust algorithm was not showing neither the best results in time nor the quality, but it was chosen as it is one of the few methods using the Parametric approach. The last methods, Screened Poisson and SSD, were selected as in [8] and [9] both methods showed similar or better results than Poisson, both in time and quality of the reconstruction.

The selection has also been based on methods with open source implementations or already developed in the Point Cloud Library (PCL), which perfectly adapts to the Telepresence platform. Fast Triangulation and Marching Cubes have an implemented function in PCL, so we had to develop a program that computes these methods using the provided functions. Then, Poisson, Screened Poisson, SSD and 3D Reconstruction are developed by [21], and they provide the source code. The source code for the Power Crust can be found in [22].

3.3 DATABASE

To perform the evaluation, we have built a database with 13 models which have different resolutions and levels of detail. The selected models must have a Ground Truth that will be the reference to evaluate the quality of the reconstruction. Each model has been resampled 3 times obtaining different resolutions: 100%, 75%, 50% and a 25% of the original points. To resample the models, we have used the filter “Point Cloud Simplification” from MeshLab. With the different resolutions, we will have the possibility to determine how the 3D Surface Reconstruction methods behave when the point cloud has higher or lower density of points.

The database has been prepared to adjust to the input requirements of each algorithm. One of the requirements from most of the algorithms, except for Power Crust, was the need to have an oriented point cloud, which means that they need the normal vector estimation to the surface. The normal

estimation has been computed using the "Normal Estimation for Point Sets" filter from MeshLab. Another difference is that Power Crust needs a different input file format than the other algorithms. Most of the algorithms need a PLY[1] file as an input whereas the Power Crust needs a file PTS, which describe point clouds only using the XYZ coordinates.

Table 1 shows the number of points of the models in the database and Table 2 shows an image of the Ground Truth mesh. The database models are taken from the Stanford Repository [23] and Aim@Shape repository [24].

MODELS	**RESOLUTIONS (number of points)**			
	100%	**75%**	**50%**	**25%**
Armadillo	172974	123982	82148	39725
Budda	543652	404535	289616	133933
Budda2	757490	567091	379197	189297
Bulldog	501500	374286	266144	120781
Dragon	3609600	2703566	1822990	909420
Frog	197170	147603	101722	49012
Gargoyle	863210	643083	401500	219927
Grog	876150	653771	441608	218935
Horse	1104470	826018	553138	276136
Neptune	2003932	1528032	1014639	503574
Ramsses	826266	613531	417494	209548
Raptor	1000080	760196	505148	252284
Statuette	4997267	3577960	2430038	1284785

Table 1: Resume of the database. We show the different models and the number of points for the different resolutions

Database Models

				
Armadillo	**Budda**	**Budda2**	**Bulldodg**	**Dragon**

[1] Previously defined in the Abbreviations

				
Frog	**Gargoyle**	**Grog**	**Horse-Isidore**	**Neptune**
				
Ramsses	**Raptor**	**Statuette**		

Table 2: Ground Truth Models from the database

3.4 EXPERIMENTS

We have performed two experiments to evaluate the algorithms:

1. The first test consists on applying the reconstruction methods to the database and then evaluate the behaviour in terms of the quality and the computing time of the reconstruction. Then, we will have to reach a compromise between both the quality and the processing time to be able to implement the algorithm into the Telepresence platform, so it may be in real-time.
2. The second test consists on applying a pre-processing to the point cloud to remove noise before computing the surface reconstruction. With this second experiment, we will study the increase/decrease of processing time compared to the quality of the mesh.

   The applied pre-processing filter is the "Outliers Statistical Removal" from PCL which removes isolated points. It analyses if the mean distance from a point *p* to *n* neighbours is larger or shorter than *d* standard deviation of the mean distance. If it is larger, the point will be marked as outlier. We have set *n=20* and *d=2*.

3.5 QUALITY MEASUREMENTS

The evaluation of the quality of the reconstruction will be measured both quantitatively and qualitatively.

3.5.1 Quantitative Measurement

The quantitative or objective quality will be measured with the Root-Mean-Square (RMS) error. The RMS, based on an approximation of the Hausdorff Distance, calculates the distance between 3D surfaces represented with triangular meshes.

If we first define a distance between a point p and a surface a S' as

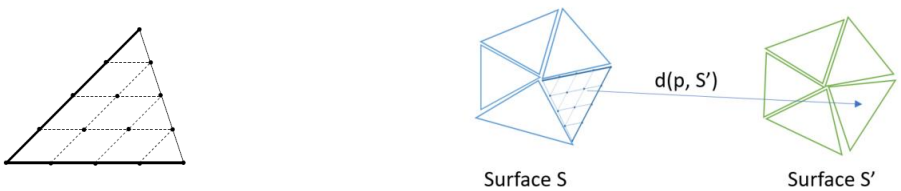
$$d(p, S') = \min_{p' \in S'} \|p - p'\|_2 \quad (1)$$

where p belongs to a surface S, then, the Hausdorff Distance between two surfaces, S and S', can be defined as:

$$d(S, S') = \max_{p \in S} d(p, S') \quad (2)$$

It is noticeable that $d(S, S') \neq d(S', S)$, where *d(S, S')* is called forward distance and *d(S',S)* backward distance. For this reason, the symmetrical Hausdorff Distance is computed as the maximum between both measures.

The process followed to compute the one-sided distance is to sample a triangle T from a surface S. This sampling is done by selecting two sides of the triangle and sample each side with n points. Using the directions of the sides, a regular grid is built according to the sampling pattern: $n(n+1)/2$. Then, equation (2) is computed, where p are the generated points. The information is more detailed in [25].



a) *b)*

Figure 6: a) Sampling grid of T for 5 points [25] b) Scheme of the process to calculate the RMS

Later on the section, we will refer to S as *sampled surface* and S' as *target surface*.

We have calculated the RMS using the Hausdorff Filter from MeshLab, which only computes the one-sided distance.

3.5.2 Qualitative Measurement

The qualitative or subjective quality of the reconstruction will be measured by applying the algorithms into real data from the Telepresence platform and decide which one performs a better surface reconstruction. Note that this is a subjective measure, so it will not have a numerical classification.

3.6 AUTOMATIZATION

The process to evaluate the different 3D Surface Reconstruction methods has been completely automatized.

3.6.1 Algorithm execution

We have developed bash scripts which execute the different algorithms for every model in the database and store the processing time in text files, as well as the resulting reconstruction in PLY files, except for the Power Crust, which saves the files in OFF[2] format. For the second experiment, we have also created a script that computes the "Statistical Outliers Removal" filter for all the point clouds in the database and stores the computing time in a text file as well as the cleaned cloud.

To accomplish an efficient execution, we have used Array Job, which is a feature from SLURM[3] that offers a mechanism for submitting and managing collections of similar jobs quickly by executing them in parallel. Additionally, Array Jobs saves the log for each job so we can see the errors in the execution. The inputs of the bash scripts are the paths to the models in the database and the location where the results will be stored. To define these inputs, we have created a "parameter file" with the path to each model and the location and output name where the resulting surface reconstruction will be stored. Then, the bash scripts read these "parameter file" and uses it as inputs for the executions.

As said before, the Power Crust algorithm behaves different than the others and it needs a different input format. For this reason, we had to implement a C++ program that convers the PLY format to a PTS file. Also, the implementation from this algorithm could not be executed with Array Job and we had to develop a different script, which computed the reconstructions sequentially.

This automatization allows us to avoid repetitive tasks achieving a more efficient execution. Moreover, in case of having to repeat an experiment the process would be faster.

[2] Object File Format. Defined in Abbreviations.

[3] SLURM is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters.

3.6.2 RMS execution

We have also automatized the process for computing the quantitative results. To compute the RMS, we have used meshlabserver, a server provided by MeshLab which allows a command line computation of its filters. We have generated a script that computes the Hausdorff Distance. This script describes the filter in XML and saves it in the MLX format. After generating the MLX script, we have created a bash script that executes the MeshLab script for all the generated reconstructions.

Note that as the Hausdorff Distance filter from MeshLab only calculates the one-sided distance, we had to generate a MeshLab script where the Hausdorff Distance is applied two times, but inverting the inputs. This is: we first apply the Ground-Truth model as sampled surface and the computed reconstruction as target surface and then, the ground-truth is the target surface and the computed is the sampled surface. After we have obtained the forward and backward Hausdorff Distances, the maximum between both measures is computed and stored in a text file.

3.6.3 CSV creation

Finally, we have created a CSV file for the data to be treated. For this, we have developed a script that collects all the data from the text files, which only have the numerical values for the computation times and the RMS values, filtered when creating the text files.

For the first experiment, we have a file with the name of the model and its resolution, the processing time and the RMS value. For the second experiment, we add the pre-processing time as well as both, the reconstruction time and the total time.

CHAPTER 4. RESULTS

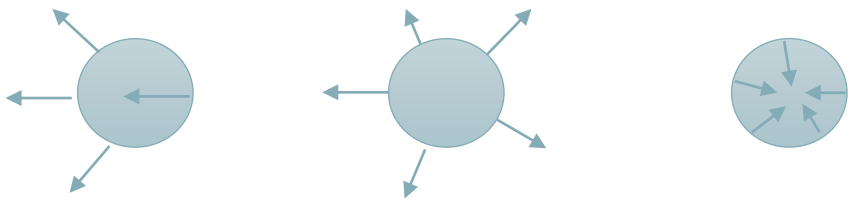
In this section, we are going to explain the obtained results after applying the 3D Surface Reconstruction algorithms to the database. We are going to analyse the results from both experiments and decide which method or methods adapt better to the Telepresence platform.

First, the potential problems with the 3D Surface Reconstruction algorithms will be studied and a comparison between the quality and execution time of the reconstruction will be done. As the models of the database have different number of points, the processing time has been normalized to compare the models. After studying the first experiment, we will compare it with the results from the second to determine the performance of the cleaning filter. Finally, we will make a conclusion to decide the suitable 3D Surface Reconstruction method for our system.

The experiments for this project have been conducted on the GPI servers based on an Intel(R) Xeon(R) @ 2.40GHz using 32GB of RAM and 2 CPUs. Notice that some of the algorithms need more memory than others and we have stablished 32GB as a standard. Also, some implementations allow concurrent execution, so some will use the 2 CPUs. The implementations using a concurrent execution are Poisson, Screened Poisson, SSD and 3D Reconstruction.

4.1 FIRST RESULTS

As we have seen in previous sections, most of the algorithms need the normal vector to the surface to determine the curvature of the surface. The normal estimation can be done following different approaches as seen in Figure 7:



a) *b)* *c)*

Figure 7: a) Normal Estimation towards one specific direction b) Normal Estimation outwards c) Normal Estimation inwards

One approach for estimating the normal vectors is to compute them outwards or inwards the object whereas the other procedure estimates the normal vectors towards a viewpoint, which forces all the normal vectors towards the same direction, like Figure 7.a. The main problem with the second

approach is that some normal vectors will be directed outside the object and others will be inwards the object.

Some of the algorithms are very sensitive to this problem and when the normal vectors are directed both inwards and outwards the object, the resulting reconstruction creates strange artefacts or surfaces where there are no points like shown in Figure 8. To solve the problem, we have calculated the centroid of the object to stablish this position as the viewpoint towards all the normal vectors will be pointing at. With this procedure, the artefacts disappear.

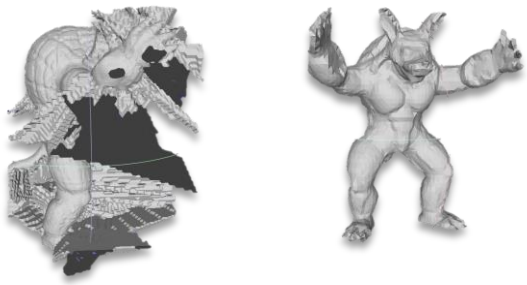


Figure 8: Marching Cubes reconstruction of the armadillo model with PCL normal estimation (Left) and MeshLab normal estimation (Right)

4.2 First Experiment

For this experiment, we will first discuss a general overview on how the algorithms behave for the different models in the database. Then, we will analyse the Time-RMS relationship and we will qualify the reconstructions from the Smart Room's data.

4.2.1 General Overview

After applying each 3D Surface Reconstruction algorithm to the database, we have seen that Power Crust has had problems with two models, "Statuette" and "Grog", where it could only reconstruct the 25% resolution. Power Crust is an algorithm that really approximates the result to the Ground Truth models, but it requires a lot of memory and time to work properly. In the case of these two models, the reconstruction could not be done for lack of memory. If we analyse both models, "Statuette" has the larger number of points in the database (5.000.000) and it also has lots of details, converting this model in one of the more difficult to reconstruct. On the other hand, "Grog" has a lower number of points (900.000), but it also has an elevated level of the detail, needing more memory to be reconstructed. The models can be seen in Table 2.

4.2.2 Quantitative Evaluation

In this section, we are going to analyse each algorithm according to the computing time and the quality of the reconstructed mesh. As said before, the models have different number of points so, to

compare the models, both the time and RMS have to be normalized. The normalization of the RMS is computed when applying the filter, as MeshLab provides the values with respect to the diagonal of the bounding box (BBox) of the mesh, see Figure 12, so the given RMS does not have a specific unit, but it is measured with respect to the Bbox diagonal.

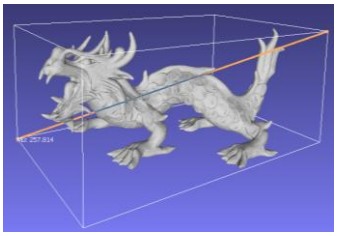


Figure 9: Dragon Bounding Box. Diagonal = 257.814

To analyse the results, we have computed the mean execution time and the mean RMS for the four resolutions, for each algorithm of the 13 different models. We have generated a graphical representation with the Time and the RMS values for each surface reconstruction method and for the models. This is the representation which better allows us to interpret the data. Nonetheless, all the graphical representations used to examine the data can be seen in the Annex B.

Now, some of the RMS-Time representations for some of the relevant models are shown in Figure 12. The graphic represents the RMS values for each resolution of the model compared with the execution time for the seven algorithms. The x-axis is in a logarithmic scale and the time is normalized.

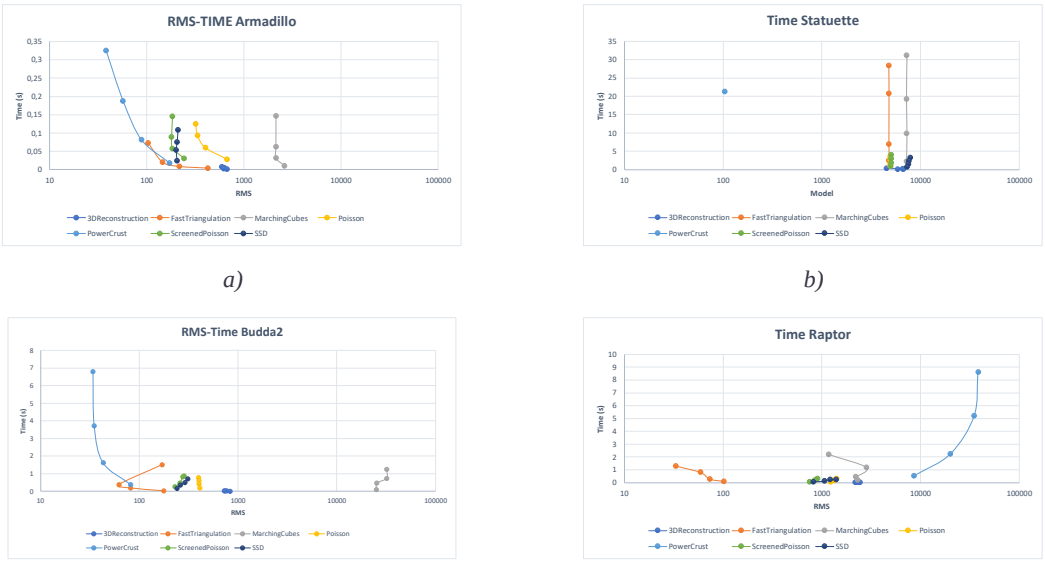
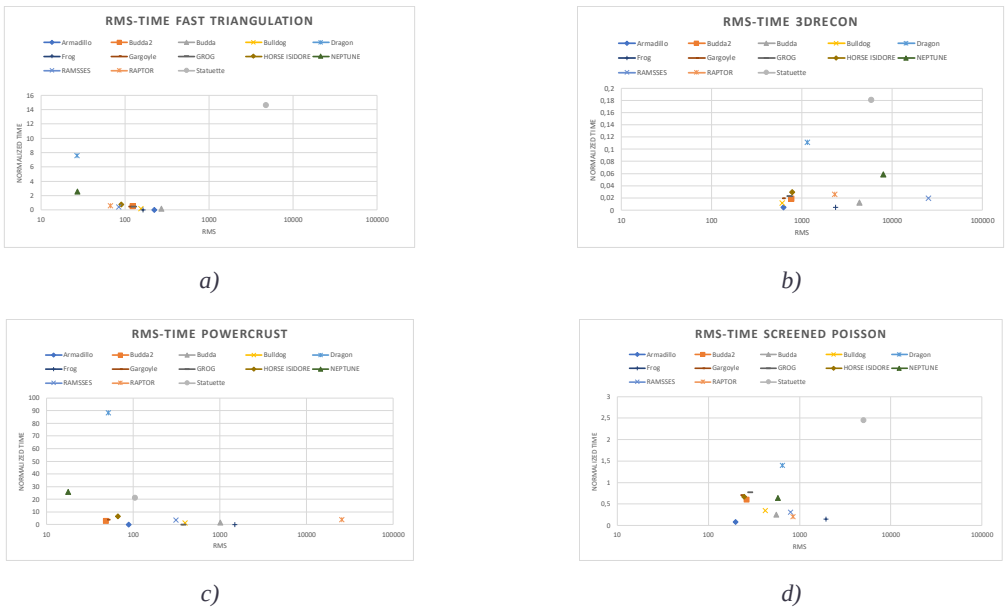


Figure 10: a) Armadillo, the algorithms behave standardly b) Statuette, where Power Crust only has one measure and the RMS values for the rest of the algorithms are close to 10000 c) Budda2, the algorithms perform standardly and d) Raptor, the Power Crust has the highest RMS and the only algorithm with a fairly good RMS is Fast Triangulation.

For a more detailed view of the algorithms' performance, some of the RMS-Time relationship for some of the algorithms is shown in Figure 11. The RMS and the time values are computed as the mean the values for the four resolutions of each model.



a) *b)*

c) *d)*

Figure 11: a) Fast Triangulation, most models are below 1s and around 100 RMS, b) 3D Recon, most models are below 0,12s, but the RMS is always higher than 500, c) Power Crust, most reconstruction are below 1000 RMS and below 100 and the time varies from less than 10s to 90s, d) Screened Poisson, most models are below 1s and the RMS is between 100 and 1000

Finally, a short conclusion for each reconstruction algorithm is presented.

- **3D Reconstruction:** This algorithm is the fastest to compute the reconstruction, approximately 1.7 seconds per model. The drawback is that the reconstruction quality is always between the worst two. The surface provided by this algorithm is too smooth and it does not reconstruct the high frequencies, which would provide a more detailed reconstruction.
- **Marching Cubes:** it is one of the slowest algorithms and the quality of the reconstruction is the poorest, where the RMS is higher than the rest of the algorithms for every model. This happens because the resolution of the meshes is not good enough and, also, it does not close properly some of the models creating big holes, which greatly increases the RMS.
- **Fast Triangulation:** this method has a low RMS for almost all models, except for the "Statuette". Also, the execution time is nearly always the second faster. It increases the computing time in the models: "Statuette", "Raptor", "Horse-Isidore" and "Neptune", which have many points.
- **Power Crust:** this algorithm generally performs the best reconstruction but with the biggest execution time. The exception is in the "Raptor" model, where it cannot complete the surface. However, the execution time is too elevated, with a mean of 330 seconds per model.

- **Poisson:** this method does not highlight neither in the quality nor the execution time. It has a similar performance to the Screened Poisson but with higher RMS and similar times comparing model by model.
- **Screened Poisson:** it has, practically in each model, the third best RMS with a medium executing time of 37 seconds. The algorithm softens the high frequency areas increasing the RMS values.
- **SSD:** it performs similar to Screened Poisson, but it is a bit faster. On the contrary, models with high complexity, like "Raptor", "Dragon" and "Neptune" have higher RMS.

4.2.3 Qualitative Evaluation

The subjective evaluation has been done by using a Point Cloud of the complete capture of the Smart Room and three foreground people, extracted using the foreground/background segmentation of the platform (see 5.2).

If we analyse the reconstruction from the different algorithms, we see that the implementations from 3D Reconstruction, Power Crust and Marching Cubes cannot support colour data. Moreover, the Power Crust does not reconstruct the object with a minimum quality due to its underperformance when working with noisy point clouds. Marching Cubes has problems with the normal estimation of the point set and it creates strange planes around the reconstructions, not allowing to see the actual shape of the model. 3D Reconstruction is better than the last ones but it does not completely preserve the form of the model.

If we analyse Poisson, Screened Poisson and SSD, which had a similar quantitative performance, it can be stablished that Screened Poisson provides a more accurate reconstruction than the other two algorithms. Poisson eliminates some regions of the point cloud and SSD creates surfaces outside de object to eliminate all possible holes. These three algorithms always try to build a watertight surface, which leads to the creation of unwanted surfaces that appear due to how the normal estimation is computed. As the normal estimation is calculated towards the centre of the object, when this object has a concave form, as Model 3 in Figure 13, both SSD and Screened Poisson create erroneous surfaces towards that point. When the object is convex, all the normal vectors would be directed inwards the object and this problem would not appear.

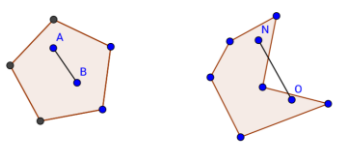


Figure 12: Convex polygon (left), Concave polygon (right)

Finally, the Fast Triangulation is not as sensitive as Screened Poisson and SSD to the normal estimation approach and the final reconstruction has a good approximation to the shape of the object. However, it does not build a complete watertight surface.

In the reconstruction of the Room, see Figure 14: Smart Room Reconstruction. a) Fast Triangulation b) Screened Poisson c) SSD, the problem with the normal estimation can be seen for the Screened Poisson and SSD algorithms and Fast Triangulation creates erroneous surfaces due to the noisy data.

In the previous section, we saw that Fast Triangulation, Screened Poisson and SSD where the algorithms with the best performance regarding the compromise between quality and time. In the subjective measurements, they also have the best reconstructions. In Figure 13, we show the reconstruction of three foreground people from the data captured in the Smart Room and Figure 14 shows the reconstruction from the whole room.

a) *b)* *c)*

Figure 13: Model 1 (left), Model 2(right), Model 3 (center) a) Fast Triangulation b) Screened Poisson c) SSD

a) *b)* *c)*

Figure 14: Smart Room Reconstruction. a) Fast Triangulation b) Screened Poisson c) SSD

4.2.4 Conclusions

To sum up, analysing the quantitative evaluation some of the algorithms are clearly not suitable for our system. Although Power Crust is one of the methods that better reconstructs the database models, the execution time is not appropriate for the requirements of the Telepresence system. Also, it fails to reconstruct the data captured in the Smart Room.

Figure 15: Model 1 (left) Model 2 (centre) Model 3(right)

Other algorithms that can be rejected are the 3D Reconstruction and Marching Cubes. Despite 3D Reconstruction is the fastest algorithm, the reconstruction quality is too poor where the minimum RMS is 600 and many reconstructions are above 1000. When the RMS values are higher than 1000, the reconstruction misbehave in some areas of the model whereas lower RMS values fail in the precision, see Figure 16. Marching Cubes has a minimum RMS of 600 and it is the second slowest algorithm so, both the execution time and the quality are not good enough. Another method that can be discarded is Poisson because its performance is like Screened Poisson and SSD but both have better RMS results.

Figure 16: Left Raptor 3DRecon 2300 RMS, Right Raptor Screened Poisson 842 RMS. Left Horse 3DRecon 800 RMS, Right Horse Screened Poisson 245 RMS

In consequence, Fast Triangulation, Screened Poisson and SSD are the best algorithms. Fast Triangulation performs the reconstruction faster than the other methods but the quality is highly sensitive to the point cloud density. This could be a problem when applying the algorithm to the Telepresence data as the point clouds are noisy and not uniformly distributed across the cloud. On the other hand, Screened Poisson and SSD are more constant and robust against nonuniform density of points. The difference between these two methods is that SSD is a bit faster than Screened Poisson but it does not reconstruct properly models with important level of detail.

For the qualitative evaluation, the best reconstruction for the Telepresence data corresponds to the algorithms with the best quantitative performance. The SSD and Screened Poisson had a similar

performance, but for the room's data, SSD generates more surfaces outside de object, although both have problems with the normal estimation.

For these reasons, Fast Triangulation and Screened Poisson are the two best algorithms. Fast Triangulation is faster than Screened Poisson and it better preserve the form of the object, but it does not build a complete watertight object, whereas Screened Poisson does.

4.3 SECOND EXPERIMENT

In the second experiment, we aim to analyse how the algorithms behave when a pre-processing is applied to the point cloud. We also want to evaluate how the computing time increases or decreases when we apply this processing.

For the pre-processing of the point clouds, we have applied an Outliers Removal filter, which eliminates sparse points from the cloud.

4.3.1 General Overview

After cleaning the point clouds and applying the algorithms to the database, we have seen that the results are worse than in the first experiment. This happens because when applying the filter to remove outliers, it eliminates relevant points of the point clouds, leading to holes in the reconstructions.

4.3.2 Quantitative Evaluation

With the previous problem, we have compared the mean RMS for each algorithm with the first experiment. In Figure 17 we can see that the RMS is worst for every algorithm, in exception for the Marching Cubes where the mean RMS does not change. This is not strange as the RMS values for this algorithm where already very elevated.

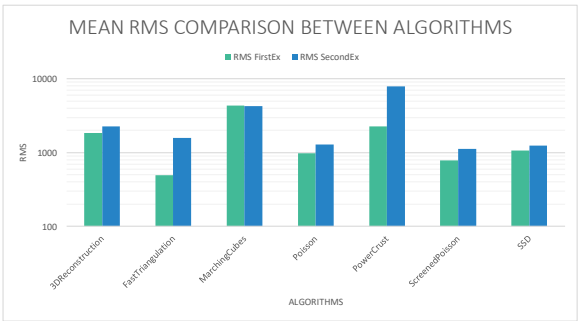


Figure 17: RMS comparison between experiments

The graphic shows that some of the algorithms are more sensitive to the point cloud quality. With the cleaning, the uniform density of the point cloud disappeared and we see that Power Crust and Fast Triangulation are the algorithms where the quality of the reconstruction has decreased more in

the new conditions. This is a crucial factor as the points clouds of the room are far from having a uniform distribution of points.

As the cleaning filter has worsen the point clouds, we will not evaluate the algorithms in further detail in the quantitative evaluation but in the qualitative.

4.3.3 Qualitative Evaluation

The qualitative evaluation will be done by comparing the obtained results for the first experiment and the second experiment. We will only consider the Fast Triangulation and Screened Poisson algorithms as we have stablished that they performed the better reconstructions both in quality and time.

	Fast Triangulation		**Screened Poisson**	
	First Experiment	Second Experiment	First Experiment	Second Experiment
Model 1				
Model 2				
Model 3				
Room				

Figure 18: Comparison between First and Second Experiment

In Figure 18, we see the obtained reconstructions comparing both experiments. For the first model, the reconstruction of the Fast Triangulation eliminates noise, especially in the head and arms. On the contrary, Screened Poisson only removes noise from the head. For the second model, in both algorithms, the shoes are worse reconstructed after applying the pre-processing stage, where data has disappeared. The third model does not improve in neither of the algorithms as Fast Triangulation eliminates part of the leg and the chair and Screened Poisson does not practically change.

Where most difference can be seen is in the complete room, as Fast Triangulation delivers a nearly perfect reconstruction, removing the noise and Screened Poisson does not close the object but does not reconstruction correctly the person inside, who has a sphere surrounding it.

For the obtained results, focusing more in the foreground models, the reconstruction does not improve after applying the pre-processing. The parameters for deciding if a point is an outlier should be more precise or even a different type of pre-processing should be implemented, for example a filter which uniformizes the data, as these point clouds do not have a uniform density of points.

4.4 EVALUATION

To conclude, we have seen that Fast Triangulation and Screened Poisson are the algorithms that best suit the Telepresence platform requirements. The first, performs a faster reconstruction but it does not completely close the object whereas the second, is slower but it always builds a watertight surface. Fast Triangulation is more sensitive to a non-uniform density of points but is less susceptible to the normal estimation calculation. On the contrary, Screened Poisson's reconstruction quality is highly dependent on how the normal vectors to point sets are calculated, but less sensitive to ununiform distribution of points.

For these reasons, we will integrate the Fast Triangulation on the Telepresence platform as, although it does not build a watertight surface and it is more sensitive to ununiform point clouds, it is faster than Screened Poisson and it is not sensitive to the normal estimation, which is a critical point on the surface reconstruction.

One of the objectives of the project was to implement the system in real time, but with the obtained results we have seen that it is not possible for the moment, as the fastest reconstruction for a foreground object is 1,5 s.

	First Experiment				Second Experiment			
	Model 1	**Model 2**	**Model 3**	**Room**	**Model 1**	**Model 2**	**Model 3**	**Room**
Fast Triangulation	1,9 s	1,6 s	2,2 s	64,0 s	1,5 s	1,6 s	1,6 s	309,7 s
Screened Poisson	21,4 s	17,8 s	18,0 s	27,7 s	21,0 s	16,1 s	18,6 s	320,3 s

Table 3: Execution time for the First and Second Experiment

CHAPTER 5. INTEGRATION

In this chapter, we are going to detail how the selected 3D Surface Reconstruction algorithm, Fast Triangulation, has been implemented into the Telepresence platform. As said in section 0, the 3D surface reconstruction section will be integrated in the "Analysis" block of the Telepresence baseline.

5.1 PLATFORM

The Telepresence platform is built over the Robot Operating System (ROS), which is a distributed framework designed for writing robot software running on Unix-based platforms. It provides services as low-level device control, message-passing between processes and package management. The runtime communication between processes follows the peer-to-peer (P2P) network model. P2P is a decentralized communication in which every process (nodes in ROS syntax), can act as both client and server.

ROS has three levels of concepts (1) Filesystem (2) Computation Graph and (3) Community Level.

1. Filesystem defines the resources.
    a. **Packages**: main unit for organizing the software. It contains nodes, libraries, datasets, configuration files, etc...
    b. **Manifests**: defines the metadata about the package.
    c. **Message type**: defines the data structure for the message sent in ROS. The messages are defined as: *package_name/MessageName.h*
    d. **Service type**: defines the request and response data structure for services in ROS.
2. Computation Graph defines the P2P communication of ROS.
    a. **Nodes**: process that perform a computation.
    b. **Messages**: the communication between nodes is done with messages, which is a data structure comprising type fields.
    c. **Topics**: messages are routed via a transport system with publish/subscription semantics. The topic is where some node publishes a message and it is used to identify such message. There may be multiple concurrent publishers and subscribers to a given topic.
    d. **Services**: is how the request/reply interactions are executed in ROS. Services are defined by a pair of messages, one for the request and one for the reply.
    e. **Bags**: format for saving and playing ROS message data. Bags are an important mechanism for storing data, such as data coming from depth sensors.

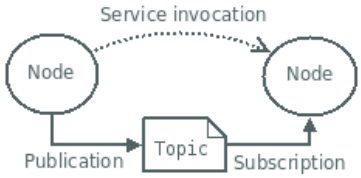


Figure 19: Computation graph structure [27]

5.2 DEVELOPMENT

The structure of the Telepresence platform is shown in Figure 20: Telepresence Platform The baseline system captures the Smart Room with three Kinects and publishes the resulting Point Cloud. Then, a transmission Node, subscribes to the Point Cloud and transmits the data over TCP/IP to the HTC Vive glasses using a C++ library for I/O programming called Boost [26]. In the analysis block a Foreground/Background segmentation Node is implemented, so only the foreground or background is transmitted.

One of the objectives of the project is to integrate a 3D Surface Reconstruction algorithm to the Telepresence platform. This integration has been done in the Mesh Node, where the Fast Triangulation method is calculated and the reconstructed surface is published. Then, the reconstruction can be visualized with the VR device.

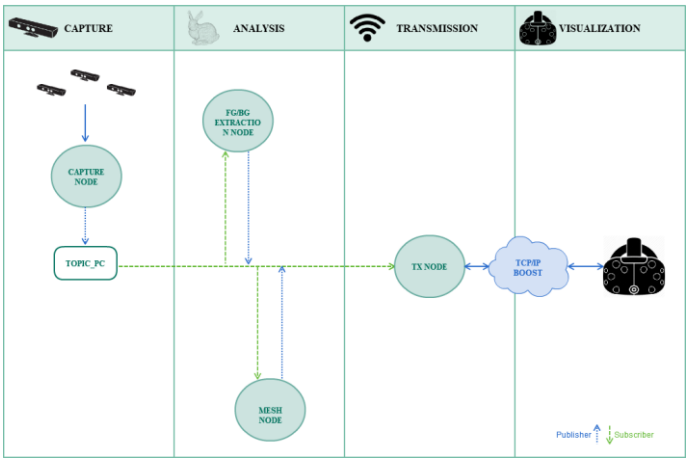


Figure 20: Telepresence Platform

5.2.1 Mesh Node

The Fast Triangulation method as well as the pre-processing filters and the normal estimation are implemented in PCL and as ROS has PCL integrated in its system it is easily integrated with the platform.

We have created a Node which subscribes to a Point Cloud and computes the normal estimation and the Fast Triangulation algorithm. It can also apply a processing filter to the point cloud. Once the surface has been reconstructed the data is published in the topic: "*topic_mesh_publisher*". The specifications of the node are as follows:

- **Subscriber**: it subscribes to the point cloud from a *sensor_msgs/PointCloud2*, which is easily converted to a PCL Point Cloud, as ROS already has a function to convert to and from PCL [27].

- **Pre-processing**: As the Fast Triangulation worsen the results when the Statiscal Outliers Removal filter was applied, see section 4.3.2, we have implemented the Moving Least Squares filter as pre-processing, which leads to better reconstructions.
    - **Moving Least Squares**: it resamples the point cloud data to have a more uniform density of points and corrects the errors due to the multiple scan registration. The search radius is set to *0.03*.
- **Normal Estimation**: it computes the surface normal towards a stablished viewpoint. We calculate this point as the centroid of the object.
- **3D Surface Reconstruction**: it computes the Fast Triangulation algorithm. For each point p, a k-neighbourhood is selected by searching for the k closest neighbours in a sphere with radius $r = \mu \cdot d_0$ where $d_0$is the distance of p to the closet neighbour and μ specifies the maximum distance for a point to be considered. Then, the points are projected to an approximately tangential plane to the surface formed by the k-neighbours and finally, the points are connected to p and consecutives points forming the edges of the triangles to build the polygon mesh. We have set *r=15,* $\mu = 3$ *and k=100*.
- **Publisher**: it generates the message that will be published. We have design a custom message as ROS did not have a message that supported colour meshes. The definition of the message will be explained in 5.2.3.

The code for publishing or subscribing to the message will be explained in 5.2.4.

5.2.2 TX Node

The Telepresence platform has a node for transmitting the data from the room to the VR glasses. This node is needed as the processing is done in Linux and the HTC vive run on Windows. The node incorporated the transmission of meshes but it was implemented to read the data from file. We have improved this implementation by adding the code for subscribing to a mesh message.

5.2.3 Message Definition

ROS has a standardised message for describing meshes. This message is from the package *shape_msgs* but it does not support colour data. The message is composed of a list of triangles, defined in the message *shape_msgs/MeshTriangle.h*, where the triangle is defined as groups of 3 vertices. Then, the points are also stored in an external message, *geometry_msgs/Point.h*, which is an array of float with the X, Y, Z position of each point.

To add the colour information, we have created a message on the package where our code is stored, called *pc2mesh*. The message is called *pc2mesh/Mesh.h* and it follows the same structure as the *shape_msgs/Mesh.h* but adding the array of colours defined as uint8.

pc2mesh/Mesh.h

```
# list of triangles; # the index values refer to positions in vertices[]
MeshTriangle[] triangles

# the actual vertices that make up the mesh
geometry_msgs/Point[] vertices

# Definition of the colour for each point
pc2mesh/Colors[] colors
```

Shape_msgs/MeshTriangle.h

```
# Definition of a triangle's vertices
uint32[3] vertex_indices
```

geometry_msgs/Point.h

```
# This contains the position of a point
float64 x
float64 y
float64 z
```

pc2mesh/Colors.h

```
# This contains rgb colour for each point
uint8 r
uint8 g
uint8 b
```

For the TX_NODE to be able to read this type of data, the message has to be imported to that package, which means that the configuration file for the packet has been modified with the following lines:

```
Find_package (pc2mesh)
catkin_package (CATKIN_DEPENDS pc2mesh)
include_directories (include ${pc2mesh_INCLUDE_DIRS})
target_link_libraries (${pc2mesh_LIBRARY})
```

5.2.4 Publisher and Subscriber

To publish the 3D surface reconstruction, the data has to be adapted to the ROS message structure by extracting the cloud, the polygons and the colour information from the data. Then, the ROS message can be built.

The integrated algorithm, Fast Triangulation, is part of the PCL library, so the format for the mesh is *pcl::PolygonMesh*. As said before, ROS supports this library and it already has some functions to extract the desired data. Once the cloud, the colour and the polygons are extracted, we have to generate the message. The *pc2mesh/Mesh.h* message can understand any type of mesh, which means, that if we wanted to use a 3D reconstruction algorithm not implemented in the PCL library it could also be integrated.

For the subscription, we have to perform the inverted adaptation where the input data is the ROS message and it must be converted to the *pcl::PolygonMesh*, as it is the required type of data used int the TX_NODE to send through boost. The subscription code will work independently of the 3D surface reconstruction algorithms used, as it already has the structure of the message.

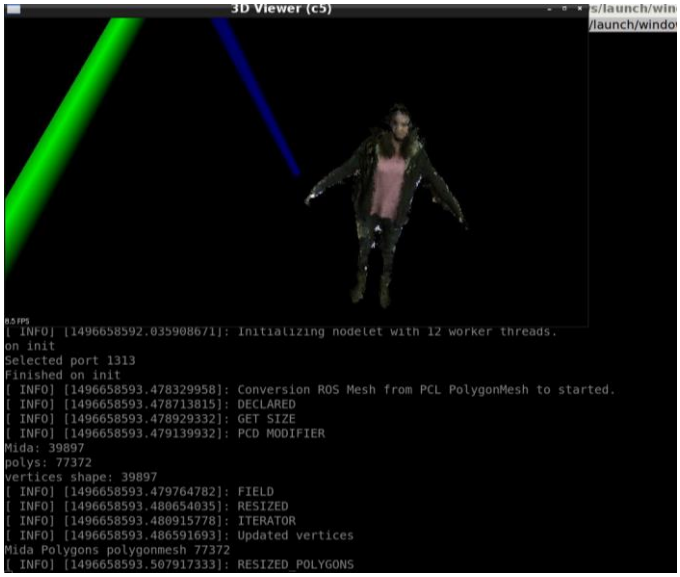


Figure 21: Mesh read in the TX_NODE before sent through boost

To summarize, we have implemented a node which reads a *sensor_msgs/PointCloud2* message and computes the 3D surface reconstruction. The data, can be either read live or from a bag file with stored data from the room. Then, the reconstruction is published with the message *pc2mesh/Mesh.h* which improves the standardized mesh message from ROS: *shape_msgs/Mesh.h*. Finally, the TX_NODE subscribes to the mesh message and transmits the data to the HTC vive glasses.

CHAPTER 6. BUDGET

In the following section, a detailed budget of the project will be presented.

To develop the project, we have required the following equipment: 3 Kinects, HTC Vive and a computer. We have used the GPI servers to perform the tests for the evaluation of the 3D Surface Reconstruction algorithms and the integration of the selected algorithm into the Telepresence platform. This includes approximately 250h for the evaluation and 130h for the integration. For this reason, we have estimated the cost per hour by using the Amazon Web Service (AWS) as a cloud computing option.

Equipment	Units	Price / unit (€)	Total Cost (€)
Kinect	3	195	585
HTC Vive	1	899	899
Computer	1	1032	1032

.

Equipment	Unit	Memory (GiB)	Price / hour (€/h)	Hours	Total Cost (€)
Server	1	32	0.46	380h	174,8
			Total equipment cost:		**2691**

Table 4: Total equipment cost. [Data Accessed: 12 June 2017]

The project has been developed for 4 months with a total of 720 working hours, computed for a 24-credit subject. The salary has been computed following the UPC agreement for internships which is 8€/h. For the tutor, the base salary of a titular professor at the UPC has been applied, 35000€/annual, which results in 20€/h, working 37,5 hours/week. In the following table, we detail this information:

Staff	Hours/week	Cost (€/h)	Working hours (h)	Total cost (€)
Internship engineer	35	8	720	5760
Tutor	1,5	20	24	480
			Total equipment cost:	**6240**

Table 5: Total staff costs

To sum up, the total budget of this project is:

Equipment	2516€
Staff	6240 €
Total Budget	**8931 €**

Table 6: Total Budget of the Project

CHAPTER 7. CONCLUSIONS AND FUTURE DEVELOPMENT

In this thesis, we aimed to study the State of the Art 3D surface reconstruction algorithms to determine which was the most suitable for the Telepresence platform, where the quality is equally important to the computing time. The results showed that few algorithms provide a reasonable compromise between both restrictions and all algorithms perform a better reconstruction with high density of points. Moreover, a critical point of most algorithms, computing the reconstruction using an implicit function approach, is the normal estimation to the point set. Achieving a good estimation of the normal vectors have proven to have a significant impact on the quality of the reconstruction. The results also showed that implementing a 3D surface reconstruction to a video sequence in real time is not possible, as for computing the reconstruction of a foreground person with approximately 40.000 points needs 1.5 seconds for the Fast Triangulation algorithm. Nonetheless, it could be applied to a previously stored video and played offline.

The second objective to accomplish during the project was to integrate one of the algorithms into the Telepresence platform. We have implemented the Fast Triangulation algorithms as it fulfilled the requirements of the platform, a compromise between both the quality and the computing time. Furthermore, we have improved the standardized mesh message from ROS to support colour information, which not only can be used by the Fast Triangulation algorithm but any 3D reconstruction algorithm. The transmission node has also been improved as it previously read meshes from stored files and now it subscribes to the new message for describing 3D surface reconstructions.

Future improvements of the project would be to implement the Screened Poisson algorithm to the Telepresence platform as it also provided a good compromise between the quality and the execution time. Moreover, to enhance the reconstruction of this algorithm, a better normal estimation could be calculated, where all the normal vectors where directed inwards the object.

Another improvement would be to apply a spatio-temporal coherence in the reconstruction, which could not be accomplished in this thesis, and should improve the reconstruction quality when applied to a video sequence.

BIBLIOGRAPHY

[1] "Siggraph 2015," [Online]. Available: http://s2015.siggraph.org/attendees/immersive-realities-arvr-contest.html. [Accessed 2017 June 28].

[2] Joachim Neumann, Josep R. Casas, Dusan Macho and Javier Ruiz Hidalgo, "Integration of audiovisual sensors and technologies," *Springer-Verlag London Limited,* pp. 15-23, 25 April, 2007.

[3] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Trans. on Visualization and Computer Graphics,* vol. 5, pp. 349-359, 1999.

[4] B. Curless, M. Levoy, "A volumetric method for building complex models from range images," *Computer Graphics (SIGGRAPH '96),* p. 303–312, 1996.

[5] N. Amenta, S. Choi, and R. K. Kolluri, "The power crust," *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications,* p. 249–266, 1987.

[6] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *In ACM SIGGRAPH Computer Graphics,* vol. 21, pp. 163-169, 1987.

[7] M. Kazhdan, M. Bolitho and H. Hoppe, "Poisson surface reconstruction," in *In Eurographics Symposium on Geometry Processing*, 2006.

[8] M. Kazhdan and H. Hoppe, "Screened Poisson surface reconstructio," *ACM Trans. Graph 32, 3,* no. 29, pp. 1-13, 2013.

[9] F.Calakli and G. Taubin, "SSD: Smooth Signed Distance Surface Reconstructio," *Computer Graphics Forum,* vol. 30, no. 7, 2011.

[10] Z. Marton, R. Rusu, M. Beetz, "On Fast Surface Reconstruction Methods for Large and Noisy Datasets," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA),* 2009.

[11] M. Kazhdan, "Reconstruction of Solid Models from Oriented Point Sets," in *Eurographics Symposium on Geometry Processing*, 2005.

[12] H. Edelsbrunner and E. Mücke, “Three-Dimensional Alpha Shapes,” *CM Transactions on Graphics,* vol. 13, no. 1, pp. 43-72, 1994.

[13] Weisstein, Eric W, “Voronoi Diagram,” From MathWorld--A Wolfram Web Resource, [Online]. Available: http://mathworld.wolfram.com/VoronoiDiagram.html. [Accessed 12 June 2017].

[14] J.-D. Boissonnat, “Geometric structures for three-dimensional shape reconstruction,” *CM Trans. Graphics 3,* p. 266–286, 1984.

[15] S. Izadi, A. Kowdle, S. Escolano, C. Rhemann, P. Davidson, S. Fanello, “Fusion4D: Real-time Performance Capture of Challenging Scenes,” SIGGRAPH ’16 Technical Paper, Anaheim CA, July 24-28, 2016.

[16] F. Prada, M. Kazhdan, M. Chuang, A. Collet and H. Hoppes,, “Motion Graphs for Unstructured Textured Meshes,” SIGGRAPH ’16 Technical Paper, Anaheim CA, July 24-28, 2016.

[17] D. Casas, C. Richardt, J. Collomosse, C. Theobalt and A. Hilton, “4D Model Flow: Precomputed Appearance Alignment for Real-time 4D Video Interpolation,” *Pacific Graphics,* vol. 34, no. 7, 2015.

[18] P. Huang, N. Tejera, J. Collomosse and A. Hilton, “Hybrid Skeletal-Surface Motion Graphs for Character Animation from 4D Performance Capture,” *ACM Transactions on Graphics,* vol. 34, no. 2, p. Article 17, 2015.

[19] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, “MeshLab: an Open-Source Mesh Processing Tool,” in *Sixth Eurographics Italian Chapter Conference, page 129-136,* 2008.

[20] “PCL - Point Cloud Library (PCL),” [Online]. Available: http://pointclouds.org/.

[21] M. Kazhdan, “Michael Mish Kazhdan,” Department of Computer Science, Johns Hopkins University, Baltimore, [Online]. Available: http://www.cs.jhu.edu/~misha/. [Accessed 13 June 2017].

[22] “Powercrust Software,” Computer Scrience Department, UCDAVIS, [Online]. Available: http://web.cs.ucdavis.edu/~amenta/powercrust.html. [Accessed 13 June 2017].

[23] Stanford, “The Stanford 3D Scanning Repository,” Standford Computer Graphics Laboratory, [Online]. Available: http://graphics.stanford.edu/data/3Dscanrep/. [Accessed 13 June 2017].

[24] “Aim@Shape. Digital Shape WorkkBench,” [Online]. Available: http://visionair.ge.imati.cnr.it/ontologies/shapes/viewmodels.jsp. [Accessed 13 June 2017].

[25] N. Aspert, D. Santa-Cruz, T. Ebrahimi, “MESH: Measuring Errors Between Surfaces Using the Hausdorff Distance,” *IEEE International Conference in Multimedia and Expo (ICME),* vol. 1, pp. 705-708, Lausanne, Switzerland, August 26-29, 2002.

[26] C. M. Kohlhoff, “Boost.Asio,” Boost Software License, Version 1.64, 17 April 2017. [Online]. Available: http://www.boost.org/doc/libs/1_62_0/doc/html/boost_asio.html. [Accessed 2017 June 23].

[27] ROS, “ROS,” [Online]. Available: http://wiki.ros.org/pcl/Overview. [Accessed June 21].

[28] P. Cignoni, C. Rocchini, R. Scopigno, “Metro: measuring error on simplified surfaces,” *Computer Graphics Forum 17,* pp. 167-174, 1998.

[29] ROS, “ROS Introduction,” [Online]. Available: http://wiki.ros.org/ROS/Introduction. [Accessed 19 06 2017].

LIST OF FIGURES

LIST OF TABLES

ANNEX A

A.1 REVISION HISTORY AND APPROVAL RECORD

REVISION	DATE	PURPOSE
0	05/06/2017	Document creation
1	13/06/2017	Document revision
2	21/06/2017	Document revision
3	22/06/2017	Document revision
4	24/06/2017	Document revision
5	25/06/2017	Document revision
6	29/06/2017	Document revision

NAME	E-MAIL
Laura Mora Ballestar	lmoraballestar@gmail.com
Javier Ruiz Hidalgo	j.ruiz@upc.edu

	WRITTEN BY:	REVIEWED AND APPROVED BY:
Date	30/06/2017	30/06/2017
Name	Laura Mora Ballestar	Javier Ruiz Hidalgo
Position	Project Author	Project Supervisor

A.2 WORK PACKAGES

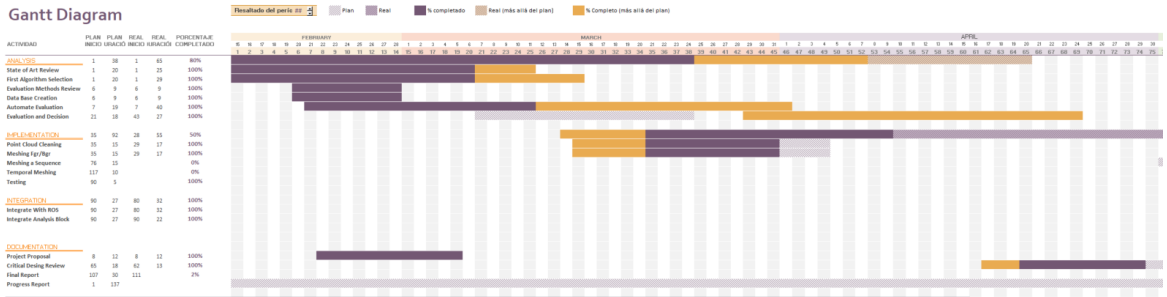
WP1	Tasks	Start date	End date
Analysis *15/02/2017* *24/04/2017*	**State of Art Review**	15/02/2017	11/03/2017
	- Review well stablished methods as well as new algorithms		
	First Algorithm Selection	15/02/2017	15/03/2017
	- From all the SoA review, select some of the Meshing algorithms to be tested in more detail		
	Evaluation Methods Review	20/02/2017	28/02/2017
	- Review State of Art methods to evaluate 3D Surface Reconstruction algorithms. - Test the evaluation methods to ascertain the performance		
	Data Base Creation	20/02/2017	28/02/2017
	- Create a DB with Ground Truth Meshes to be compared with the created ones. - For each point cloud, generate at least 3 simplifications with less points.		
	Automate Algorithm Application and Evaluation	22/02/2017	1/04/2017
	As we have a large database, we need: - A script to apply the different algorithms - A script that automatically applies the evaluation method (We have automatized both experiments)		
	Evaluation and Decision	29/03/2017	24/04/2017
	- Evaluate the obtained results and decided which is the best meshing algorithm for the Telepresence system - Evaluation regarding both experiments		

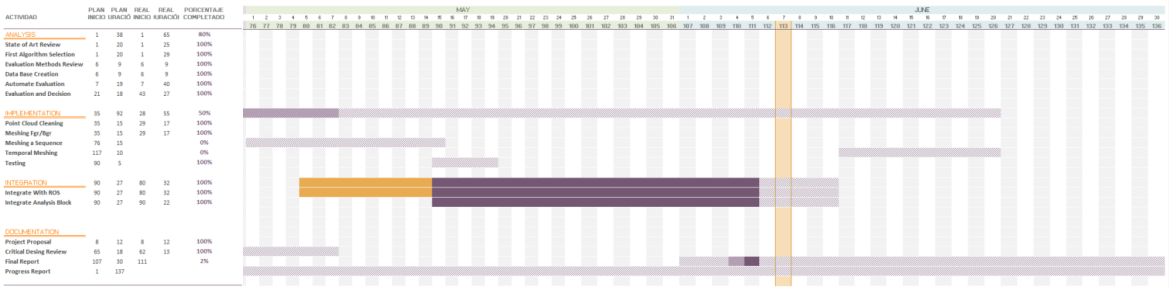
WP2	Tasks	Start date	End date
Implementation *21/03/2017* *09/05/2017*	**Meshing Algorithm into Telepresence Cloud**	24/03/2017	31/03/2017
	- Point Cloud Cleaning: the ply from the room will need some sort of pre-processing/post-processing step - Meshing Foreground and Background: decide to reconstruct both or only foreground and repeat the background.		
	Temporal Meshing	11/06/2017	21/06/2017
	- Add temporal redundancy to the meshing algorithm to reduce the computed time. - Take advantage to the temporal redundancy to smooth movement.		

WP3	Tasks	Start date	End date
Integration *08/05/2017* *30/05/2017*	**Integrate with ROS**	15/05/2017	10/06/2017
	Read and Write meshes - Create a ROS message for meshes *Stream Mesh*: adapt the transmission to ROS system - View the meshed sequence in the HTC vive glasses		
	Integrate with the Analysis Block	15/05/2017	10/06/2017

WP4	Tasks	Start date	End date
Documentation *15/02/2017* *30/06/2017*	**Project Proposal**	22/02/2017	05/03/2017
	First Description of the Project		
	Critical Design Review	20/04/2017	07/05/2017
	Revision on the state of the project		
	Final Report	01/06/2017	30/06/2017
	Final Document		
	Progress Report	15/02/2017	30/06/2017
	Document every step of the project development		

A.3 GANTT DIAGRAM

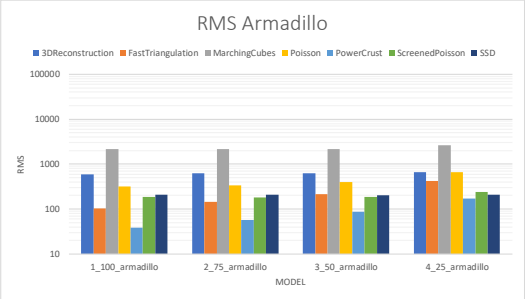


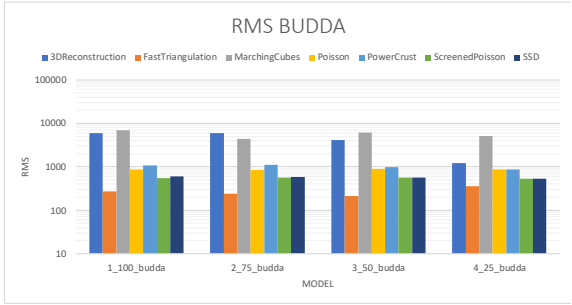


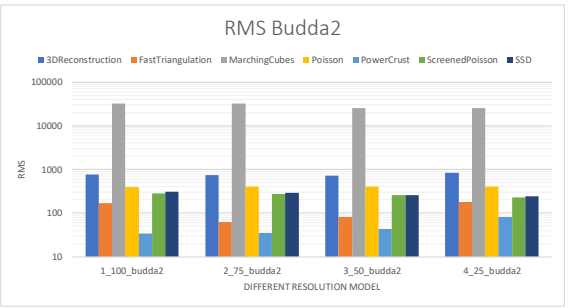
ANNEX B

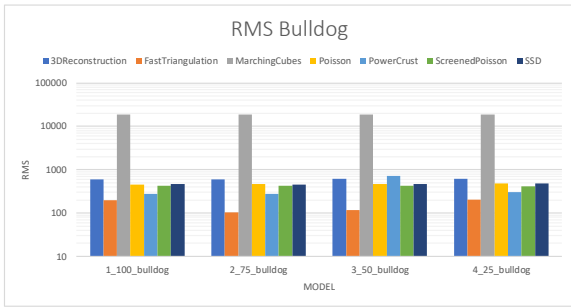
B.1 RMS REPRESENTATION

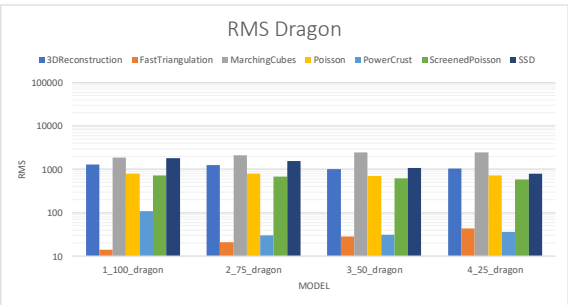
RMS graphic representation for each model, where the y-axis is in logarithmic scale.

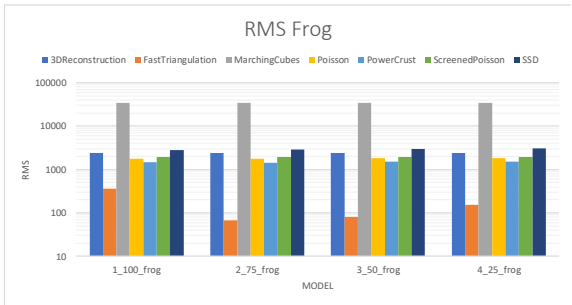


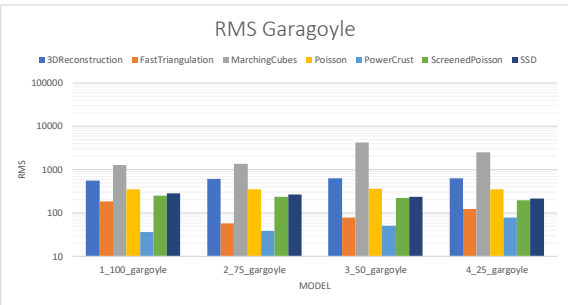


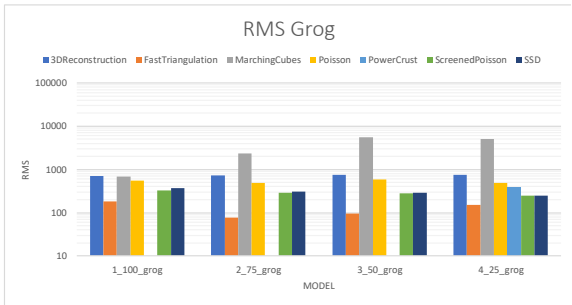


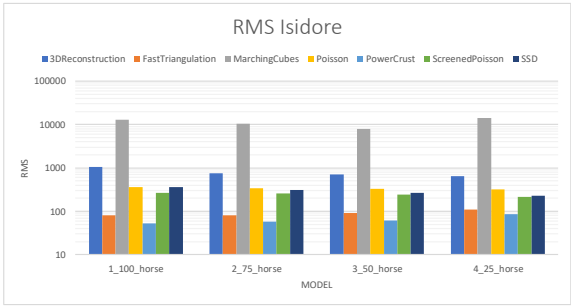


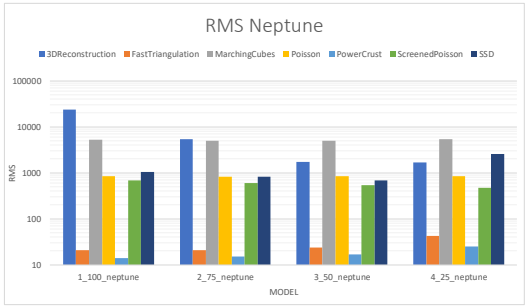


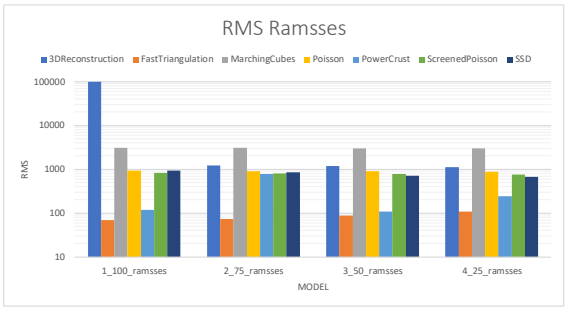


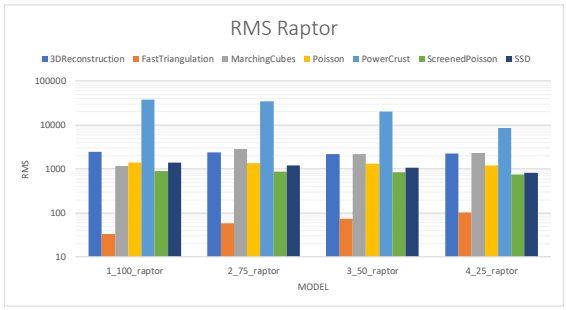


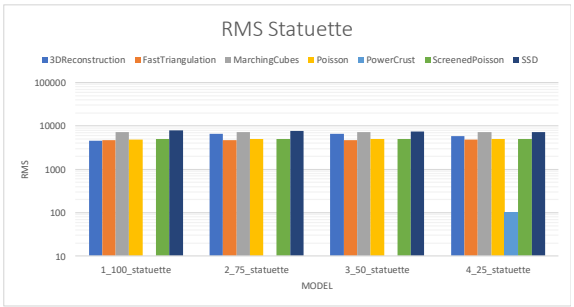



RMS Isidore
3DReconstruction
FastTriangulation
MarchingCubes
Poisson
PowerCrust
ScreenedPoisson
SSD
100000
10000
1000
100
10
RMS
1_100_horse
2_75_horse
3_50_horse
4_25_horse
MODEL


RMS Neptune
3DReconstruction
FastTriangulation
MarchingCubes
Poisson
PowerCrust
ScreenedPoisson
SSD
100000
10000
1000
100
10
RMS
1_100_neptune
2_75_neptune
3_50_neptune
4_25_neptune
MODEL

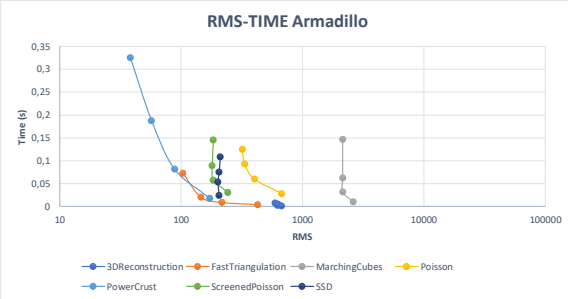

RMS Ramsses
3DReconstruction
FastTriangulation
MarchingCubes
Poisson
PowerCrust
ScreenedPoisson
SSD
100000
10000
1000
100
10
RMS
1_100_ramsses
2_75_ramsses
3_50_ramsses
4_25_ramsses
MODEL

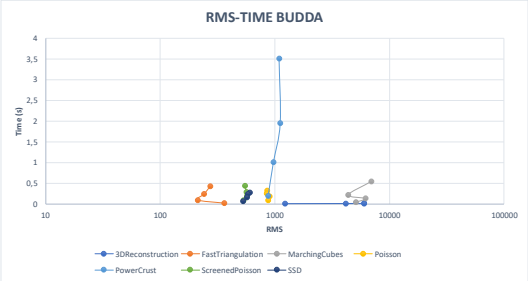

RMS Raptor
3DReconstruction
FastTriangulation
MarchingCubes
Poisson
PowerCrust
ScreenedPoisson
SSD
100000
10000
1000
100
10
RMS
1_100_raptor
2_75_raptor
3_50_raptor
4_25_raptor
MODEL

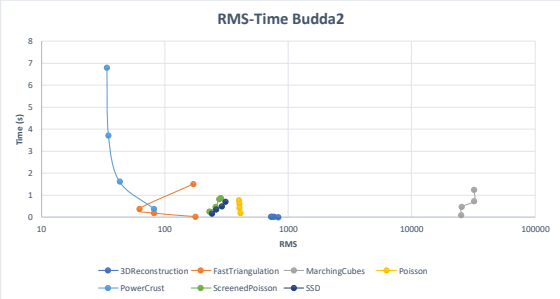

RMS Statuette
3DReconstruction
FastTriangulation
MarchingCubes
Poisson
PowerCrust
ScreenedPoisson
SSD
100000
10000
1000
100
10
RMS
1_100_statuette
2_75_statuette
3_50_statuette
4_25_statuette
MODEL

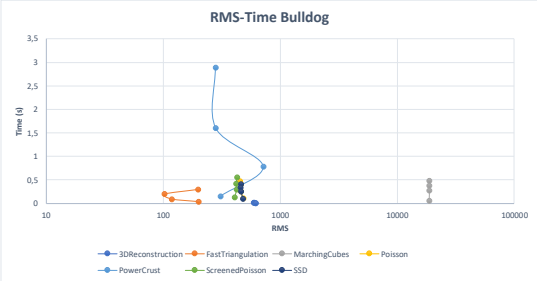
B.2 RMS-TIME REPRESENTATION FOR THE MODELS

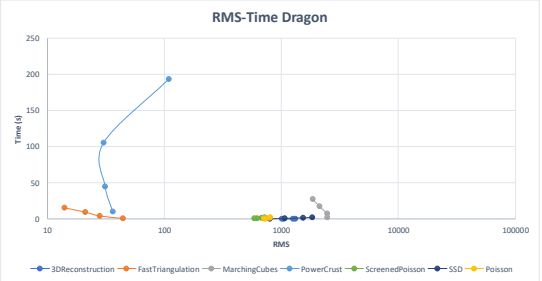
RMS-Time graphic, where the x-axis is in logarithmic scale and shows the RMS. The y-axis shows the normalized computation time for each model.

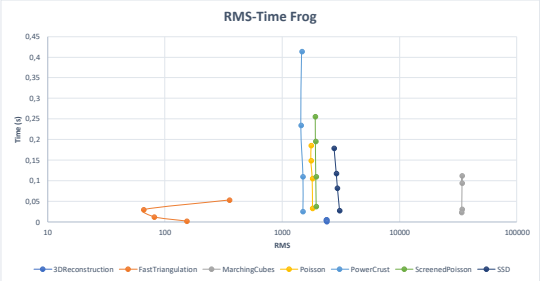


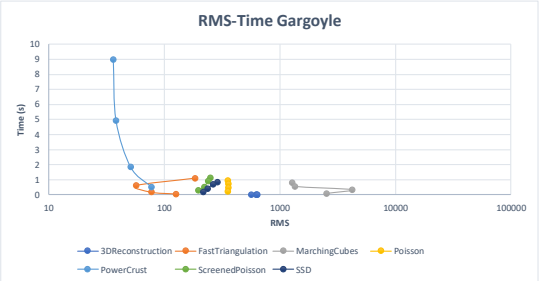


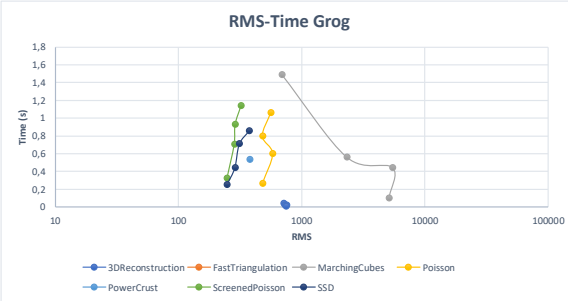


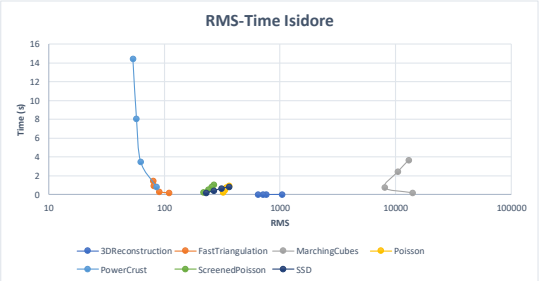


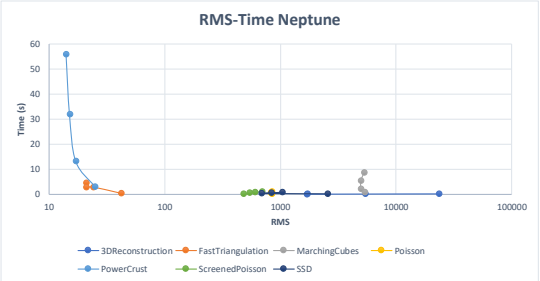


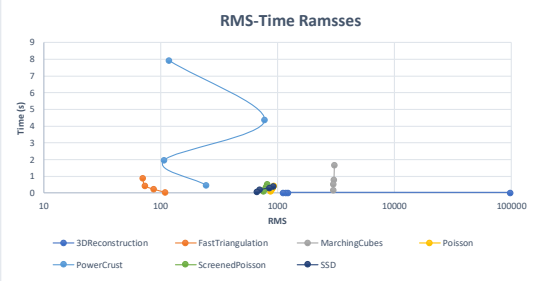


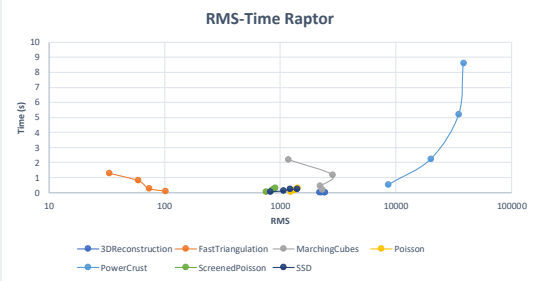


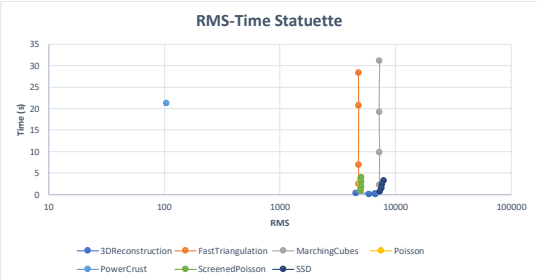







RMS-Time Ramsses
Time (s)
RMS
3DReconstruction
FastTriangulation
MarchingCubes
Poisson
PowerCrust
ScreenedPoisson
SSD


RMS-Time Raptor
Time (s)
RMS
3DReconstruction
FastTriangulation
MarchingCubes
Poisson
PowerCrust
ScreenedPoisson
SSD


RMS-Time Statuette
Time (s)
RMS
3DReconstruction
FastTriangulation
MarchingCubes
Poisson
PowerCrust
ScreenedPoisson
SSD

B.3 RMS-TIME REPRESENTATION FOR THE ALGORITHMS

RMS-Time for the different algorithms. Each point in the graphic represents the mean RMS and time for the models. The time is normalized.