THE REPUBLIC OF TURKEY

BAHCESEHIR UNIVERSITY

LOW-COST 3D RECONSTRUCTION OF OBJECTS

Master of Science Thesis

BAHTİYAR KABA

İSTANBUL, 2014

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
COMPUTER ENGINEERING

LOW-COST 3D RECONSTRUCTION OF OBJECTS

Master of Science Thesis

BAHTİYAR KABA

Supervisor: Asst. Prof. Dr. Övgü ÖZTÜRK ERGÜN

İSTANBUL, 2014

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
COMPUTER ENGINEERING

Title of the Master's Thesis : Low Cost 3D Reconstruction of Objects
Name/Last Name of the Student : Bahtiyar Kaba
Date of Thesis Defense : 01.09.2014

This thesis has been approved by the Graduate School of Natural and Applied Sciences.

Assoc. Prof. F. Tunç BOZBURA
Graduate School Director

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Science.

Asst. Prof. Tarkan Aydın
Program Coordinator

This is to certify that we have read this thesis and that we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Science.

Examining Committee Members	Signature
Thesis Supervisor: Asst. Prof. Dr. Övgü Öztürk Ergün	....
Member: Asst. Prof. Dr. Egemen Özden	....
Member: Asst. Prof. Dr. Devrim Ünay	....

ACKNOWLEDGEMENTS

I would like to express my special appreciation and thanks to my advisor Asst. Prof. Dr. Övgü Öztürk Ergün who has been a great mentor for me. I would like to thank you for encouraging my research and for allowing me to pursue this academical work. Your advice on research hase been priceless.

I would also like to deliver my special thanks to my lab partners, Hüseyin İnan and Mustafa Tosun for their valuable help at times of disaster.

A special thanks to my family and friends. Words cannot express how grateful I am to my mother, father and sisters who have always been there for support when things in life start to feel tough. This thesis study would have never succeded without them, and without the best flat-mate Ahmet Müslüm Küçük who has always been a friend in need. I also want to thank Aycan Aluçlu, Cansu Ergün and İrfan Harmancı for their great friendship and for their endless support.

Turk Telekom has provided the financial resources for this research. I also want to thank them for realizing and initiating this study.

İstanbul, 2014 Bahtiyar Kaba

.

ABSTRACT

LOW-COST 3D RECONSTRUCTION OF OBJECTS

Bahtiyar Kaba

Computer Engineering

Thesis Supervisor: Asst. Prof. Dr. Övgü Öztürk Ergün

September 2014, 67 pages

In today's computing, 3D models are a vital part of many applications from surveillance to cultural heritage preservation and most of these systems require digital representations of real world objects. Acquiring real world objects with accuracy is carried out by laser scanners which are expensive to use and they are complicated for a regular user to operate. Research is now focused on building low-cost, yet high-quality scanners with cheap range sensors.

In this thesis, we research the field of 3D scanning with commodity range sensors and propose a system design with Kinect. We modularize our work as data acquisition, registration and surface reconstruction; and evaluate corresponding algorithms on how they perform on Kinect depth output. We compare the results in accuracy, completeness and RMS metrics as well as assess the visual quality output. From the literature survey and experimental analysis that we have done, we propose a framework for 3D reconstruction with Kinect commenting on the possible improvements.

Algorithms for filtering depth images are evaluated against two datasets, one is publicly available and one is compiled by us. Bilateral filtering is the best performing when both quantitative and visual results considered. In the proposed scanner setup, the depth maps are acquired and filtered with bilateral kernel, and registered with a globally optimal graph based registration scheme which is shown to be appropriate for scanning full loops around an object. The final cloud is input to Poisson algorithm to produce a mesh representation. The effect of the octree depth size for Poisson is investigated on both visual and quantitative aspects.

Keywords: 3D Reconstruction, Kinect, Depth Scanning

ÖZET

NESNELER İÇİN DÜŞÜK MALİYETLİ 3B TARAMA SİSTEMİ

Bahtiyar Kaba

Bilgisayar Mühendisliği

Tez Danışmanı: Yrd. Doç. Dr. Övgü Öztürk Ergün

Eylül 2014, 67 sayfa

Bilişim alanında 3B modellerin yeri önem kazanmıştır ve gözetim, kültürel mirasın korunması gibi birçok uygulamada bu modellere ihtiyaç duyulmaktadır. Bu gibi uygulamalarda gerçek nesnelerin 3B dijital temsilleri kullanılmaktadır. Gerçek nesnelerin yeterli düzeyde hassasiyet ile taranması lazer tarayıcılar ile yapılmaktadır, fakat bu aletler pahalıdır ve kullanılması sıradan bir kullanıcı için zordur. Son zamanlarda düşük maliyetli ve ulaşılması kolay olan menzil sensörleri ile yüksek kalitede tarama yapılması üzerine çalışmalar olmaktadır.

Bu tez çalışmasında, ucuz menzil sensörleri ile 3B tarama teknolojileri geliştirme üzerinde araştırmalar yapılarak; Kinect sensörü ile bir tarama sistemi önerisinde bulunulmuştur. Çalışma; veri alma, örtüştürme ve yüzey oluşturma olarak bölümlendirilmiş olup, ilgili algoritmaların bu kısımlardaki performansları test edilmiştir. Sonuçlar hassasiyet, tamamlılık ve etkin değer(RMS) kritelerine göre sayısal olarak karşılaştırıldığı gibi görsel açıdan da incelenmiştir. Yapılan literatür araştırması ve elde edilen deneysel bulgular ışığında 3B tarama sistemi önerisi yapılmıştır.

Menzil imajlarını filtreleme işlemi için geliştirilmiş algoritmalar, biri halka açık biri bizim oluşturduğumuz iki veri kümesi üzerinde incelendi. Çift yanlı (bilateral) filtreleme, sayısal ve görsel sonuçlar birlikte göze alındığında en iyi sonuç veren algoritma olarak değerlendirildi. Önerdiğimiz sistemde menzil imajları taranıp, çift yanlı filtreleme ile filtrelendikten sonra elde edilen nokta bulutları yerel olmayan genel odaklı örtüştürme ile birleştirilir ve Poisson algoritması ile poligon model oluşturulur. Poisson algoritmasında octree derinliğinin sonuç üzerindeki etkisi de deneysel olarak incelenmiştir.

Anahtar Kelimeler: 3B Model Oluşturma, Kinect, Menzil ile Tarama

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

3D	: Three dimensional
ICP	: Iterative closest points
NURBS	: Non-uniform rational B-spline
OS	: Operatings system
PCL	: Point cloud library
RMS	: Root mean square

1. INTRODUCTION

The main objective of this thesis is to evaluate methods and algorithms for depth scanning and produce a 3D scanning solution in a cost effective manner. Microsoft Kinect, a low-budget range sensing device released for game consoles, is a good asset that could be integrated into such a system considering its price and availability.

A 3D model is a digital representation of an object that can be stored and processed in computers for various purposes like visualization, games and movies, quality assurance, prototyping and many more. These digital models can be crafted by artists possibly effected by their preferences and imagination, or can be scanned from real objects.

3D scanning projects are generally carried out with laser scanners which are expensive and using them requires expert level knowledge. They also require accompanying complex software. Furthermore, the laser beams emitted from these devices prevents them from being used on sensitive objects such as ancient artifacts. Their principal selling point is that the output digital models are high-quality, precise and accurate. Their error levels can be as low as tenths of millimeters.

The Kinect sensor’s output is a depth map with a resolution of 640x480. This data quality is very low compared to laser scanners, yet worse, it contains high amounts of noise. One of the goals of this work is to process Kinect data to improve our output models and minimize the error level difference between laser outputs. These enhancements can be required in different stages of the scanning procedure.

The procedure for scanning can be fundamentally separated into three parts or stages, be it for laser or Kinect(Chen and Medioni 1991, Curless 1999) :

a. **Data Reading:** The reference object is placed in front of the sensor and data is captured from around the object with varying viewing angles. The scans should be adjusted to cover all 360 degree rotation around. Filtering or superresolution methods are applied to each scan to remove noise and improve the quality.

b. **Registration (Alignment):** The scans from the previous step are generally depth maps each having their own coordinate system. To integrate them into a merged model, they first need to be transformed according to a common reference coordinate system so that the overlapping regions in different scans would align.

c. **Surface Reconstruction:** The aligned scans in the form of point clouds are merged to output a final mesh or other 3D surface representation.

For each of this stage many algorithms are proposed to augment it. In the literature survey section, we describe these methods in detail and evaluate their performance for different metrics and usage purposes.

From the knowledgebase we got from the survey, we design and propose a system solution to 3D scanning. The scanning is carried out with a Kinect sensor. The object is placed in front of the sensor on a rotating stage. 36 frames are captured each having a 10 degrees of rotation interval. Depth filtering algorithms are applied to each depth map to deal with noise and extract smoother depth maps with minimum loss in detail.

To align the depth scans, a common solution is the Iterative Closest Point algorithm (Besl and McKay 1992). It basically works by finding corresponding points in a pair of clouds and calculates the transformation matrix minimizing the error between the corresponding points. The variants of this algorithm differs in the type of error metric used, some examples of these are point to point distance, point to plane distance, color similarity (Rusinkiewicz and Levoy 2001) . ICP(Iterative Closest Point) algorithm works on pairs of point clouds, registering all clouds in a scan dataset requires aligning every adjacent pair of clouds. However, this creates a problem because the error between the first cloud and subsequent clouds accumulates towards the last cloud. This problem is referred to as Loop Closure Problem in cloud alignment (Lu and Milios 1997) . Especially the error between first and last frame is significant thus the loop cannot be closed properly. In our system design, we employ a graph based implementation of ICP (Lu and Milios 1997) that will distribute error to all transformations evenly and mitigate loop closure errors.

After the alignment has been properly done, the final clouds are merged with a meshing algorithm. We use Poisson algorithm since it can build water tight complete models and smooth out the unwanted geometry (Kazhdan et al. 2006).

Our contributions from this study includes:

a. A summarized generic framework for 3D scanning applicable to different setups. Different methods can be integrated in each step to compare and evaluate improvements.
b. A visual and quantitative comparison of filtering algorithms on depth maps.

c. Visual and quantitative evaluation of the effect of octree depth parameter in Poisson surface reconstruction algorithm.
d. A dataset of three objects in 30-60cm length. The dataset includes Kinect and Konica Minolta VI-900 scans. Laser scans can be assumed as ground truth.
e. Low-cost scanning system for Kinect which can be used to produce triangulated meshes of objects with less than 5mm error for 90 percent of vertices.

The subsequent chapters are organized as follows:

In chapter 2, we present the literature survey on the field of range scanning and 3d model reconstruction. Some preliminary information is provided which will be useful for explaining algorithms. Then, range sensors and specifications of Kinect sensor is given. What algorithms and methods are applied to improve depth map output of Kinect is discussed. The alignment or registration problem and the solutions proposed to these problems, and surface reconstruction algorithms is presented.

In chapter 3, we show our system design and present comparisons of algorithms for each stage of 3D scanning procedure. We reference some of the work presented in chapter 2 and investigate them to evaluate for 3D scanning with Kinect.

In chapter 4, we present a wider and more thorough comparisons of some algorithms with quantified results.

In chapter 5, conclusion and final remarks are given with open research questions in this area.

2. LITERATURE SURVEY

In this section, we present our research output on the field of 3D scanning. Different algorithms and methods are discussed that tries to improve the output models of 3D scanners. Even though some of them are developed in use with laser scanners, their principles might apply to scanning with Kinect-like depth cameras.

First, a general description and explanation of the common terminology used will be given for the reader far from the subject. Then, the survey will be partitioned into three parts as described in the Introduction section – Data Reading, Alignment, Surface Reconstruction- , and each section will comprise the relevant algorithms. There are also some works that deals with 3D scanning as a whole, evaluation of such systems will also be presented.

2.1 PRELIMINARIES AND DEFINITIONS

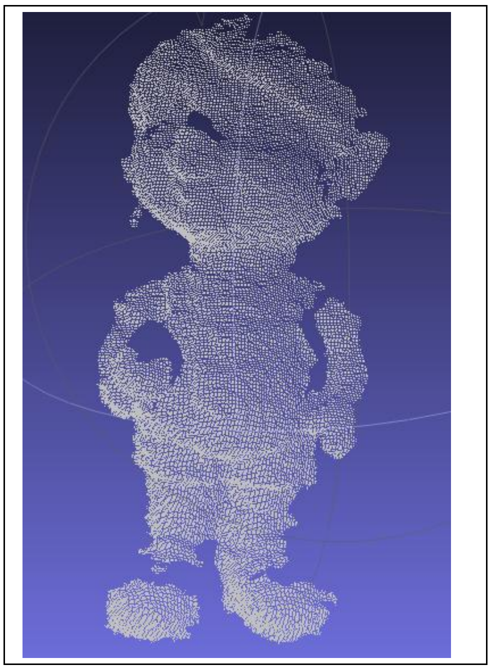
2.1.1 3D Models and Their Representations

A 3D digital model is a representation of a real or imaginary object in computers. Unlike 2D representations like color images from regular cameras, they contain the shape and geometry information. One type of such 3D models is the *Point Cloud* . A Point cloud is a collection of points defined by three dimensional vectors, and each element of a vector corresponds to *x, y, z* value with regards to a particular coordinate system. Refer toFigure 2.1 for a visual representation. Some point clouds can also contain the vertex normal, curvature and color information, in this case, the cloud vectors will be six or more dimensional.

Besides this collection of points, a model can have a set of polygons which are lists of indices to the elements in the point list. These type of representations are called *Polygonal Mesh*. This is a widely used model type, and if each polygon contains three vertices (points), it is quite useful in some tasks such as rendering since a triangle would necessarily be a plane and not a free form (Watt 1993).

Parametric Forms describes the geometry of models with mathematical equations (Campbell and Flynn 2001).

Figure 2.1: Example Point Cloud



Source: This figure has been prepared by Bahtiyar Kaba.

A generic parametric form can be defined as follows:

$$S(u,v) = \begin{bmatrix} x = f(u,v) \\ y = g(u,v) \\ z = h(u,v) \end{bmatrix} \qquad \textbf{(2.1)}$$

In the above equation, the functions f(u,v) , g(u,v) and h(u,v) are parameterized by the variables (u,v), and the surface is defined by points (x,y,z) in 3D world coordinates. From this mathematical definition, we can infer that (x,y,z) values will be in a continuous space, thus the surface is continuous one as opposed to a discrete representation in Polygonal Meshes. This is true in theory, but again, since the computers digitizes everything to work, and practically a certain operator can use discrete approximations of the parametric functions with the advantage of decreasing quantization when needed. Generally octree data structures are used to store and process such data. A common parametric formulation is Non-Uniform Rational B-Spline (NURBS) which has parameterized control points (Piegl 1991, pp. 55-71).

Algebraic implicit surfaces defines the surface as a zero set of function *f* (2.2).

$$S = \{(x,y,z) \mid f(x,y,z) = 0\} \qquad \textbf{(2.2)}$$

Equation (2.2) basically states that the surface composes the points (x,y,z) such that f(x,y,z) will be zero. Generally, for the points inside the surface f(x,y,z) will be smaller than zero and outside greater than zero. Implicit surfaces are used in the KinectFusion to build 3D models of environments (Newcombe et al. 2011) . This representation is used for mathematical computations in Poisson algorithm (Kazhdan et al. 2006).

Several other 3d object representation exist, but for the scope of this research we will not go into detail on them. However, inTable 2.1: Comprasion of 3D representation types a summary of these representations and their properties is presented. The reader can follow the source for more information.

Table 2.1: Comprasion of 3D representation types

Representation	Global	Compact	Local control	Complete	Easily sampled	Easily fit
Parametric	Yes	Yes	Yes	Yes	Yes	No
Implicit	Yes	Yes	No	Yes	No	Yes
Superquadratic	Yes	Yes	No	Yes	Yes	Yes
Cylinder	Yes	Yes	No	Yes	Yes	No
Mesh	No	No	Yes	Yes	Yes	Yes

Source: (Campbell and Flynn 2001, A survey of free form Object Representation)

2.1.2 Range Scanning

Although, some studies have focused on 3D reconstruction from color cameras and produce compelling results (Wu et al. 2011) , reconstruction from range sensors continues to be a subject of interest, because illumination variations will have less effect on them, thus making them more reliable. Reconstruction from color cameras are referred as *passive* scanning, while special range sensor which works by projecting rays are called *active* sensors.

Range sensors work by projecting light on a surface and either measures the distance by time of flight or models the distortion of a structured pattern. Kinect is a structured light scanner. From its emitter, a structured pattern of infrared light is cast. The sensor then evaluates how the pattern of light is distorted due to shape of objects in front of it, and calculates depth measurements pixel-wise. The output is depth map, that is an array of specified resolution whose each element contains a distance value.

These distance values can later be used to back-project to 3D space to find vertices given the camera calibration parameters (Szeliski 2010) . Once all the pixels in a depth image are back-projected, a point cloud is extracted as shown back in Figure 2.1. Here, it may also be necessary to apply a *background subtraction* technique to isolate the object from the surroundings.

The process of integrating scans from different view angles, called registration, and merging them are explained in the subsequent sections.

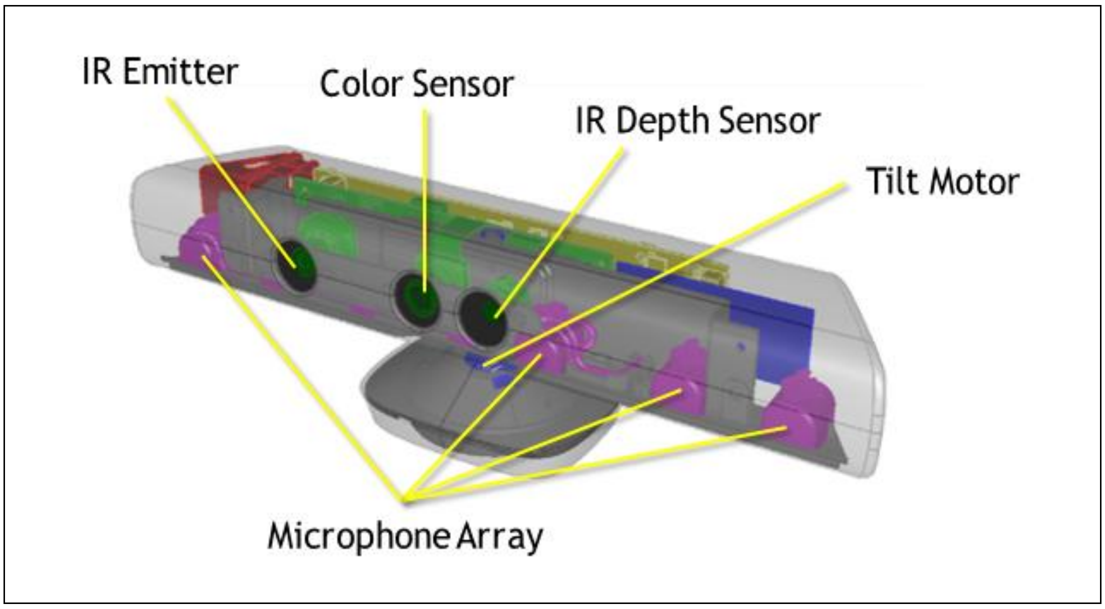
2.2 KINECT SENSOR

Kinect sensor is a range scanning sensor that is introduced for game consoles. The user can play the games by gestures without touching the controller. The console implementation process the depth values fed to it, and does skeleton tracking, gesture recognition etc.

Figure 2.2 shows the structure of Kinect. It contains an infrared emitter which will project structured light patterns and a sensor which will sense the distortions of the patterns to find the depth data.

The raw output from Kinect is an 11-bit 640x480 depth image. This depth image is very useful or 3D reconstruction because with the known camera calibration parameters of the sensor, a back-projection will produce the associated point cloud data for a depth map.

Figure 2.2: Kinect Sensor



Source: MSDN kinect website http://msdn.microsoft.com/en-us/library/jj131033.aspx

2.3 DEPTH DATA ENHANCEMENT

The output from Kinect depth sensor is highly noise contaminated and subject to systematic bias. Trying to reconstruct a 3D model with this raw input would lead to incomplete and inaccurate methods. Moreover, even if the noise had been removed completely, low-resolution of depth images will not allow a higher detail model to be built. Because of these problems, preprocessing the depth frames before aligning and reconstructing the surface is a vital part of a 3D scanning system.

We can subdivide the works focused on depth data enhancement into two categories: depth filtering and upsampling. Depth filtering is to reduce the noise and compromise the systematic bias that could have been occurred during scanning. Upsampling methods focus on generating higher quality depth images from low-resolution sometimes by using a higher-resolution color input.

2.3.1 Filtering

It has been a common practice for color imaging devices would apply one or several filtering algorithms to the raw sensor input, because the input data can be affected from illumination, hand motion and so on. The same reasoning can be applied to the depth maps, and we should also note depth cameras are more sensitive to noise due to their working principle.

First attempts to filter a depth map and remove the noise have been borrowed from the traditional image denoising algorithms.

Mean filter is a simple filter whose application field is extensive. It is effective on reducing random noise (Gonzalez and Woods 2007) .It also smoothes out the signals. However, this smoothing can have negative effects on depth map for two reasons. The smoothing is good to remove noisy data but it can also discard higher detail data, high frequency regions in the depth maps can correspond to a noise or a pointy end. Second, the smoothing of the boundary pixels results in unwanted artifacts (Vijayanagar et al. 2012). For these reasons, a direct mean filter cannot be applied to depth maps.

To deal with the boundary pixels, a weighting scheme is applied to elements of the filtering kernel (Zhao et al. 2013). Instead of the evenly distributed weights on each element, a weight is assigned which is calculated by its similarity to the center pixel. This way, for example, a boundary pixel on the object will be effect by the pixels within the object not from the pixels that belong the background even if they are adjacent pixels.

The above weighting scheme can be varied by applying different similarity metrics for giving weights to the filtering kernel. In (Zhao et al. 2013), they apply a direction aware, multi scale metric to alleviate the problems related to quantization. Multi-scale filtering derived from scale-space theory which is used widely in many filtering algorithms.Witkin(1984) proposed a multiple-scale approach to analyze the features in a digital image, since not all features are prominent in a certain scale. Generally, what is called an *image pyramid* is built by scaling down the original image down with decreasing factors. Then the features are searched in each scale giving broader definition of the image with larger number of features than that of a single-scale image. This, multi-scale scheme is also used on depth maps to detect boundaries, corners and neighboring pixels which will aid the kernel weighting algorithm for filtering (Zhao et al. 2013) .

Bilateral filtering, again first proposed for regular color or gray scale images, is another method that can be adjusted to be applied to depth maps (Tomasi and Manduchi 1998). Bilateral filtering smoothes out the noise in the image, preserving the boundaries. It works by penalizing pixel weights regarding their geometric or photogrammetric symmetry with central pixel. The application of bilateral filtering to depth maps is straight-forward since it already deals with the problems related to boundary pixels.

Kinect Fusion system makes use of the bilateral filtering to preprocess captured depth maps before applying their reconstruction method (Newcombe et al. 2011).

Some of the methods for filtering focuses on Kinect, and investigates properties for depth-denoising. Ngueyn et al.(2012) makes an experimental analysis on Kinect noise, and derives a generic model. With this generic model, they filter Kinect depth maps to remove noise. Fu et al.(2012) analyze the behavior of Kinect data, and proposes an appropriate spatial-temporal denoising algorithm.

2.3.2 Upsampling

Kinect or other consumable ToF cameras work with rather small resolutions. Kinect can produce depth frames of size 640x480, which ideally corresponds to 307200 vertices of a point cloud. However, most of this data is noisy and mostly the object we are scanning will cover only a portion of a frame. Therefore, it is not rudimentary to reconstruct a higher quality model provided the given resolution.

Upsampling methods try to increase the resolution of the depth maps while attaining noise reduction. Superresolution methods applied on color images generates higher resolution output by combining several inputs taken from slightly varying angles (Farsiu et al. 2006). A similar approach is applied on depth maps in the work LidarBoost (Schuon et al. 2009). The object is constantly spinned in front of sensor, and every 10 frames are integrated to produce a higher resolution depth map, in this manner 60 high resolution maps are captured and high quality model is produced by registering them. Subsequent works employing Lidarboost for depth denoising and upsampling proposes complete 3D reconstruction systems (Cui et al. 2010).

Color cameras can capture images in greater resolution compared to depth cameras. The color information also contains cues to the depth of the scene even if not complete. Therefore, if we have both depth and color image taken from the same view angle, which is mostly hard in practice, we can exploit color data to enhance the depth map. These types of methods are generally named *Image Guided Upsampling/ Denoising*. For such a method to work, the depth and color image should be taken from the same position, angle and calibration parameters, yet since this is unlikely, algorithms have been to develop to map the color information from a slightly displaced camera to depth maps of the depth sensor.

Chan et al. (2008) uses an image guided method to improve the quality of depth maps and to filter noise. They modify the classical bilateral filtering approach so as to work with a pair of depth and color image. High-resolution color image contains higher-frequency details that would be important for depth upsampling. This information is employed in a multi-lateral adaptive weighted filtering of the depth maps.

Park et al. (2011) uses a similar approach to exploit high-resolution color images. A regularization which applies nonlocal means filtering with an edge weighting scheme to depth maps is proposed.

Stereo methods incorporates color images from different angles to give depth estimates of the scene. Zhu et al. (2008) merges the depth output from stereo methods with time-of-flight camera outputs. Probabilistic functions are used to weight and select the correct depth value for a point. They also propose better calibration techniques for time-of-flight cameras to improve their results.

Some empirical methods have been studied that regards the Kinect output as a model, applies statistical calculations to derive this empirical model. This model is used to guide theupsampling and depth filtering for any scene (Ngueyn et al. 2012, Fu et al. 2012).

2.4 SCAN ALIGNMENT

It is necessary to scan and capture depth frames at many view angles from around the object to build a complete model since some parts of the object will be occluded in a specific frame. However, point clouds re-projected from depth frames will be defined on their own coordinate systems. One needs to apply the necessary transformations to point clouds such that they all sit in a common reference coordinate system (Hartley and Zisserman 2003) . In this case, the points from all the appended clouds will be ready to define the output 3D model.

This job is performed in two steps as *coarse alignment* and *fine alignment.* Coarse alignment is the process of roughly aligning the clouds. A simple solution is to restrict the motion of the sensor during scanning. Some predefined movement path is defined. For example, if it suffices, the object is rotated 10 degrees on certain axis between each scan, yielding 36 scans for a full capture. Since physically applying rotations exactly is not practical, that is there will be little fluctuations from 10 degrees between each frame. *Fine alignment* is applied after the clouds are coarsely aligned to minimize the transformation error.

Coarse alignment can also be achieved by finding distinct features on the objects such as corners and transforming the clouds so as to overlap these features (Makadia et al. 2006) . It can be thought possible that these features can be adequate to fine align the clouds, however, noisy and sparse nature of depth maps prevents this.

For aligning still objects, the transformations required to be applied are rigid since the form of the object is constant. Non-rigid registration techniques are used to align clouds belonging to objects that are moving during scanning (Huang et al. 2008). These algorithm are shown to be useful for reducing noise also.

2.4.1 Fine Alignment

Fine alignment is an important part of 3D scanning, because if the error between two overlapping scans is rather high, merging the points in overlapping regions will be cumbersome, and will produce low-quality output if merging threshold is lowered for it to work.

Iterative Closest Point algorithms is broad set of algorithms that are incorporated into many 3D scanning solutions(Rusinkiewicz and Levoy 2001). ICP algorithm aligns a pair of point clouds provided they have enough overlapping region and coarsely aligned. It works by finding the corresponding points in two clouds, and minimizes the distance between these points. The distance metric is not necessarily the Euclidean distance. The derivations of the ICP algorithm use different metrics and corresponding point selection methods. If the face data is available for the model cloud, point-to-plane distant metric can be used instead of a point-to-point metric.

2.5 SURFACE RECONSTRUCTION

Once all the captured information (points) is aligned correctly, they need to be merged and a surface maybe reconstructed depending on the application. In Section 2.1.1, different representations for 3D models have been discussed. The output merged cloud can be the final model, or a polygonal mesh can be produced if it is required. Parametric forms can also be created to describe the object with some predefined shapes.

For the scope of this study we focused on polyhedral mesh output, and investigated algorithms for them. Poisson is a good performing algorithm that is able to reconstruct triangulated meshes from unorganized point clouds. It also performs noise removal and smoothing to some extent (Kazhdan et al. 2006) .

Marching cubes (Lorensen and Cline, 1987) is another popular algorithm to triangulate 3D scan data which was first proposed to be used in medical imaging. Several works came after it that tries improve the results (Nielson and Hamann 1991, Hoppe et al. 1992). However, marching cubes and its derivative algorithms do not deal well with manipulating point position geometry, thus are sensitive to noise. For this reason, we cannot apply it to our strategy since we will use Kinect (high noise).

Surface reconstruction from mathematical models or point sets is a well studied problem and it has a huge application area. Numerous other works have been studied on it for different purposes like visualization, 3D data compression, transmission, physics simulations and so on. For the scope of our study, we have stipulated that we should not go into details of these studies.

3. 3D SCANNING PIPELINE

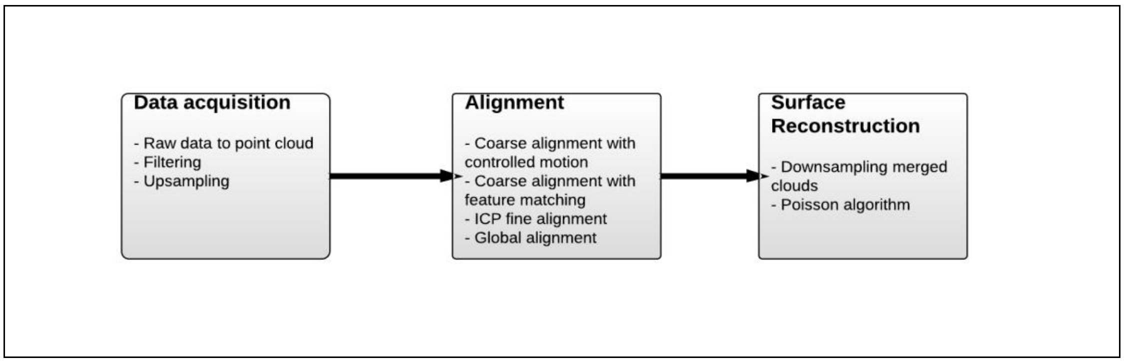
There is a pipeline of stages for 3D scanning that is used for most of the projects. As we categorized our literature survey these stages have become more clear. In this section, considering these stages, we build a 3D scanning solution as a system and describe in detail the algorithms used, and present their output.

3.1 SYSTEM DESCRIPTION

The use case for our system will be as such: The user places the object to be scanned in front of the sensor at around 70cm away. Scans are taken with 10 degrees of rotation in-between. The rotation is realized by a rotating stage which can apply specified rotations. The scans taken are aligned into a common world frame, and finally the surface is reconstructed.

A schematic representation of the components forming our system is shown inFigure 3.1 . In data acquisition part, the object is rotated in front of the sensor, and raw data is saved. This raw data is then filtered to remove noise, and back-projected to get the finer point clouds.

Figure 3.1: 3D Reconstruction Pipeline Overview


Data acquisition
- Raw data to point cloud
- Filtering
- Upsampling
Alignment
- Coarse alignment with controlled motion
- Coarse alignment with feature matching
- ICP fine alignment
- Global alignment
Surface Reconstruction
- Downsampling merged clouds
- Poisson algorithm

Source: This figure has been prepared by Bahtiyar Kaba.

3.2 DATA AND EVALUATION METHOD

Our dataset that we will use to evaluate methods in filtering, alignment and surface reconstruction consists of two subsets.

Doumanoglou et al. (2013) created a public dataset of objects scanned with both Kinect and Konica VI-700 digitizer. We used this dataset to evaluate and make numeric comparison for filtering and surface reconstruction. However, we did not use it produce full models since the scanned objects are tiny and is not very feasible for visual assessment with Kinect output.

We have also created a dataset of our own by scanning four objects which are referred as following in the text: 'boy', 'angel', 'buddha', 'three angles' . We scanned the objects with Kinect, and Konica Minolta VI 900 digitizer as ground truth.

3.2.1 Accuracy and Completeness

To quantitatively evaluate the results and compare them with ground models, there are two metrics that can be of use: accuracy and completeness(Doumanoglou et al. 2013) .

Accuracy is the defined as a distance *d* for which a certain per cent of points in the reconstructed model (from Kinect) has a nearest neighbor closer than *d* in the ground truth model. For example, if the 75 per cent of points in the reconstructed model has a distance lower than 1 cm to its nearest neighbor in ground truth model, it is said that the accuracy is 1cm with 75 per cent threshold.

Completeness measures how complete the model is built. It is calculated by percentage of points in the ground truth model which has a nearest neighbor in the reconstructed model closer than a certain distance threshold. That is, if the model is 90 per cent complete with 5mm threshold, 90 per cent of the points in the ground truth model is closer than 5mm to a point in reconstructed model.

3.2.2 RMS Error

Another metric that could be of use for evaluating the reconstructed model's resemblance to the ground truth model is to use the Root-Mean-Square (RMS) error. For calculating RMS, for each point in the cloud, the nearest neighbor in the reference cloud is found according to Euclidean distance, and this distance is stored as deviation value. In equation (3.1), Δ_i denotes the deviation of nth point, n is the number of points in the cloud. The RMS error value for a cloud is calculated as in the equation(3.1).

$$\mathrm{RMS} = \sqrt{\frac{1}{n}({\Delta_1}^2 + {\Delta_2}^2 + {\Delta_3}^2 \ldots + {\Delta_n}^2)} \qquad \textbf{(3.1)}$$

3.3 ENVIRONMENT AND TOOLS

For completing the computation tasks in this study, we used a PC with Intel i7-2600 @3.40 Ghz and 8 GB RAM with both Windows 7 64-bit and Ubuntu 12.04 64-bit installed as OS.

For 3D data processing, there are several useful, open-source software that we tried. CloudCompare[1]is a 3D point cloud and mesh visualization and comparison software with open-source license. We used it to produce visual error coded comparison between point clouds.

Meshlab[2] software has editing and processing utilities that is helpful on working and manipulating point clouds.

The system implementation has been carried on Ubuntu 12.04, because we wanted to make use of the Point Cloud Library (PCL)[3] which is easier to integrate in a Linux environment. PCL is an extensive library which is written in C++and includes implementations of many general point cloud operations such as transformations, correspondence estimation, projection, registration and surface reconstruction. We preferred a C++ implemented library for efficiency.

Matlab has been used mostly in the experimental study. For trying the filtering methods, we imported them to Matlab, and did processing there. Quantitave analysis is realized on Matlab, too.

3.4 DATA READING

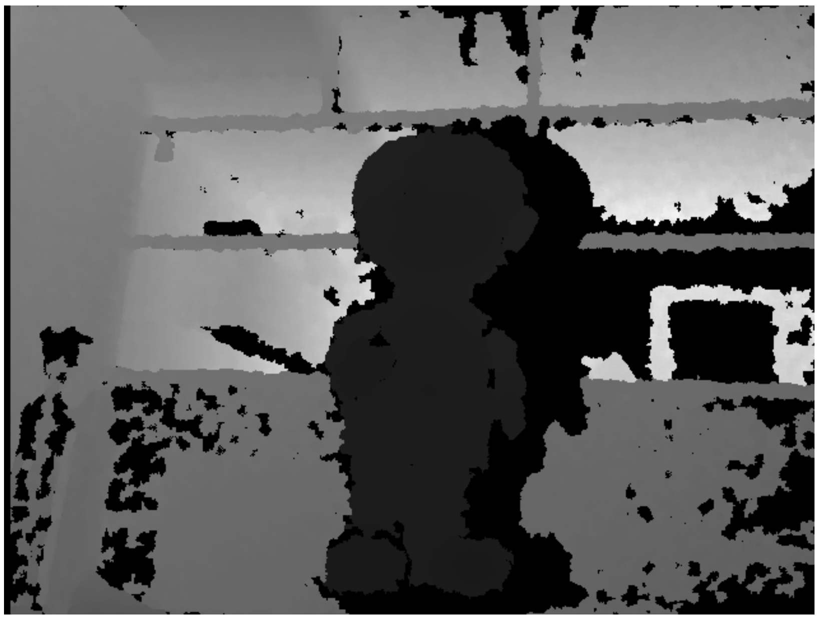
3.4.1 Depth Data

The data scanned from Kinect is an array of values that gives distance information in millimeters from the sensor. It is like a color image with pixels having depth values. A visual representation of depth map can be seen in Figure 3.2 . Here, for visualization purposes depth values are scaled to range 0-255 to display as a gray-scale image. The corresponding RGB image of the same scene is shown in Figure 3.3.

[1] Cloud Compare website. http://www.danielgm.net/cc/
[2] Meshlab website. http://meshlab.sourceforge.net/
[3] Point Cloud Library website http://pointclouds.org/

With the camera calibration parameters of Kinect (Khoshelham 2011, Konolige and Mihelich 2012, Burrus 2014, Willow Garage[4]), we can back-project the pixels to extract the point cloud for a single scan.

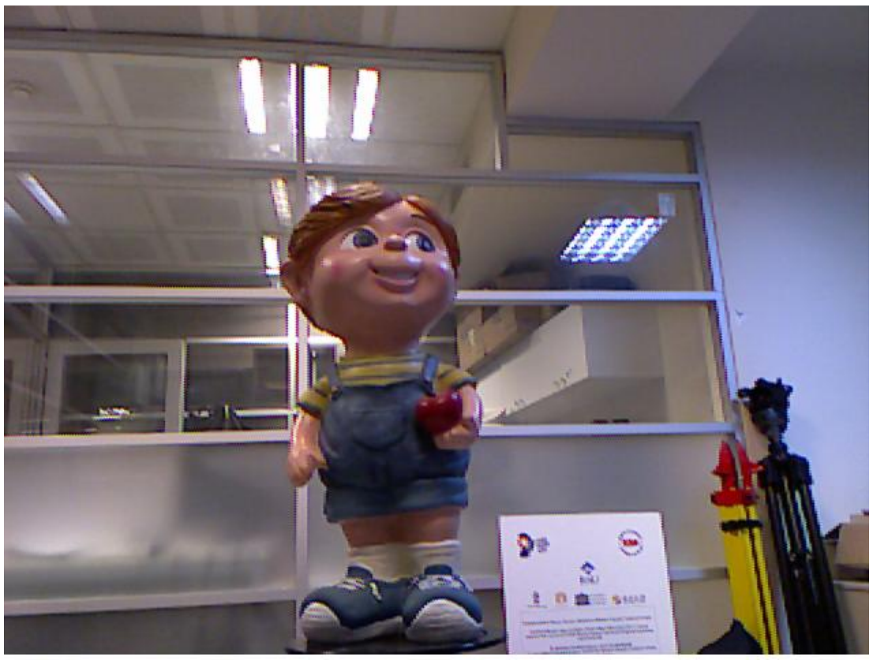
Figure 3.2: An example depth map



Source: This figure has been prepared by Bahtiyar Kaba.

[4] Willow Garage website. Kinect calibration code complete. http://www.ros.org/news/2010/12/kinect-calibration-code-complete.html

Figure 3.3: Corresponding Color Image



Source: This figure has been prepared by Bahtiyar Kaba.

The purpose of this work is focused on reconstructing objects rather than full scenes or environment, therefore it is necessary to segment the object in depth images. For this, we restrict our scenes such that the object in the foreground will be isolated well enough. This way removing the background is simpler by setting a distance threshold.

3.4.2 Background Removal

For background removal we have tried two methods. First, the scanning can be restricted such that the object will be at a certain distance from the sensor, for example between 50cm-100cm. Then, after the raw depth maps have been acquired, the pixels having depth values other than the interval 50-100cm(500mm-1000mm in Kinect) is labeled as environment. This is a simple, yet effective solution for working fast and getting results and in this study we created our dataset this way. However, for a real world scenario it can be a limiting factor.

Otsu thresholding (Otsu 1979) is an algorithm that adaptively selects an appropriate threshold for segmenting the background and foreground in grayscale images. The depth maps can also be seen as grayscale images, and therefore, if the object is isolated well from the background, a mask can be generated by binarizing the depth map with Otsu. In our system, we make use of Otsu thresholding for background removal.

3.4.3 Normal Estimation

The normal data for each vertex in a point cloud is necessary for the latter surface reconstruction step. It is possible to estimate the normals in the back-projected point cloud. But this can be computationally more expensive and sometimes normals will be ambiguous (with 180 degree). Therefore, we can exploit the depth map, and the neighborhood of pixels to estimate them.

The pixels in the depth data correspond to z-value of the vertices of the point cloud. Therefore, the cross product of the image gradient in x and y directions will give an estimate of that pixel's normal. For calculating image gradients equations (3.2) and (3.3) can be used which gives discrete approximations to the derivative of the intensity function(Gonzalez and Woods 2007, Grady and Poimeni 2010). In the equation, P(i,j) denotes the 3D vertex that is calculated by back-projecting pixel(i,j). The normal for the point corresponding to pixel(i,j) is then calculated as in equation (3.4) . It is also important to apply a smooth filter to the depth map for normal calculation because the noise can distort the results.

$$\frac{\delta I}{\delta x}(i,j) = \frac{1}{2}\left((P(i,j+1) - P(i,j)) + (P(i+1,j+1) - P(i+1,j))\right) \qquad \textbf{(3.2)}$$

$$\frac{\delta I}{\delta y}(i,j) = \frac{1}{2}\left((P(i+1,j) - P(i,j)) + (P(i+1,j+1) - P(i,j+1))\right) \qquad \textbf{(3.3)}$$

$$N(i,j) = \overrightarrow{\frac{\delta I}{\delta x}(i,j)} \; \mathbf{x} \; \overrightarrow{\frac{\delta I}{\delta y}(i,j)} \qquad \textbf{(3.4)}$$

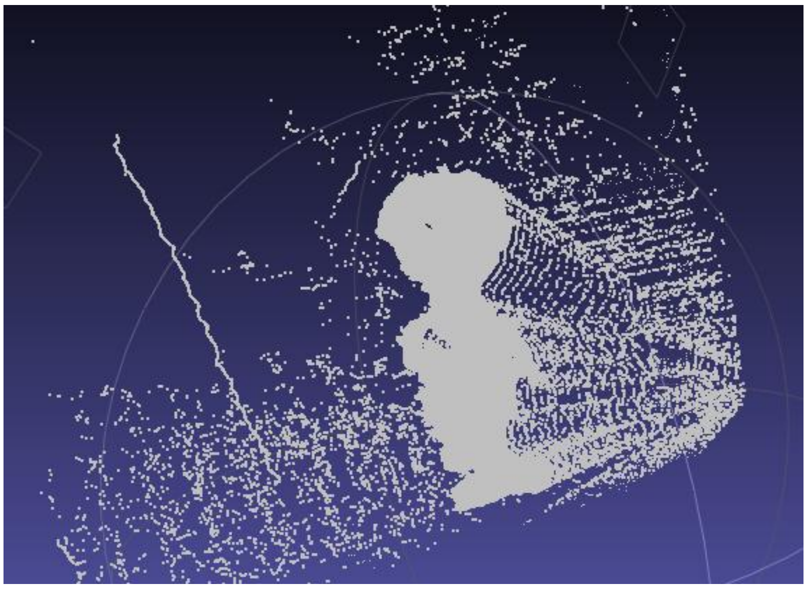
3.4.4 Filtering

3.4.4.1 Mean filtering

To remove unwanted noise from depth data, filtering methods will be applied to depth data. Mean filtering is a common method that works well with color images to deal with noise. However, directly applying a mean filtering approach to depth images is not practical since artifacts can appear along the boundaries or edges (Vijayanagar et al. 2012).

The aforementioned problem can be seen in Figure 3.4 . The pixels along the boundaries of the boy object is filtered with the pixels that belong to background and have value of zero. Even though the background pixels can be in the local neighborhood of boundary pixels, they need to be ignored for filtering since their depth values are irrelevant.

Figure 3.4: The problem of classical mean filter



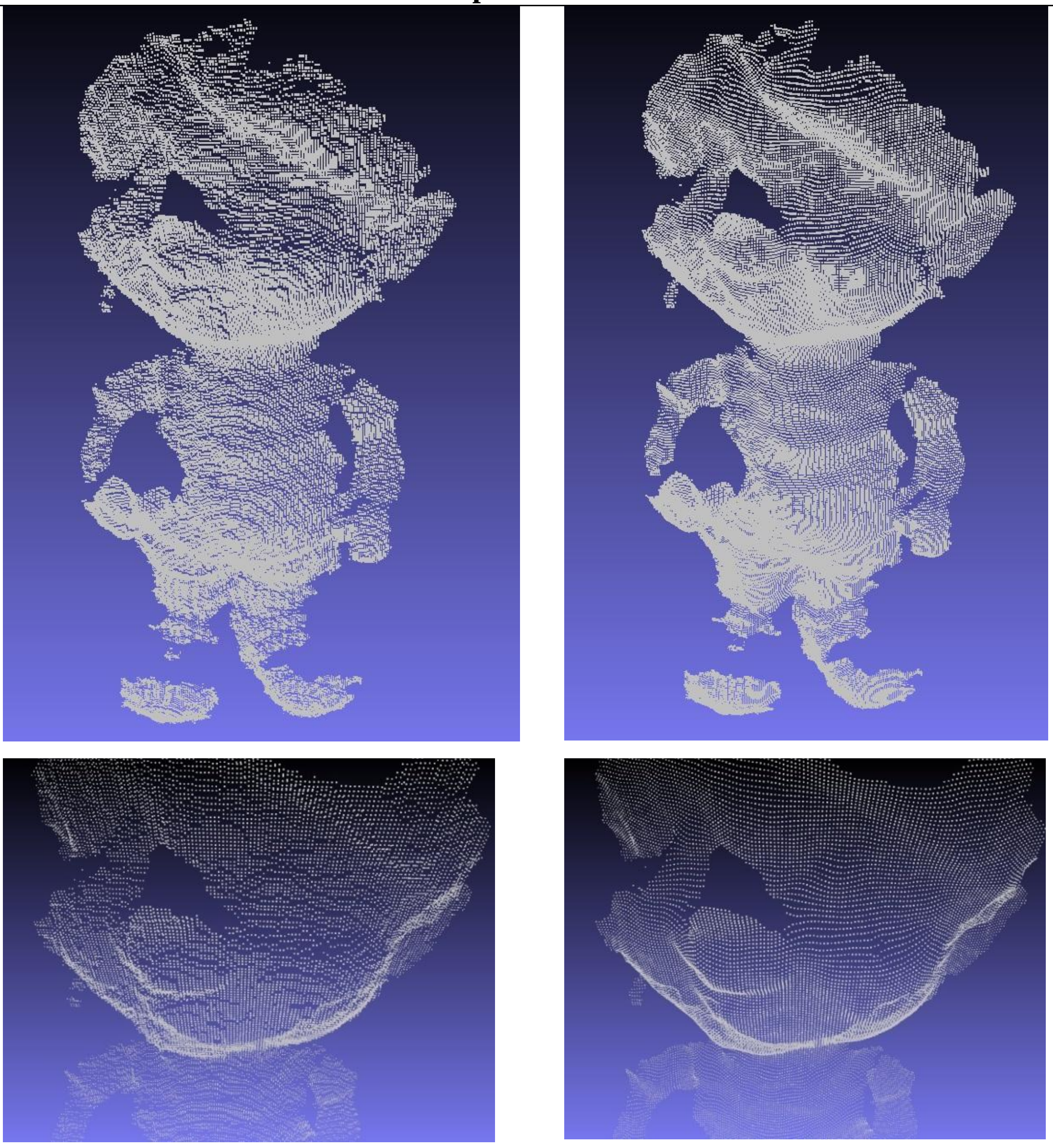
Source: This figure has been prepared by Bahtiyar Kaba**.**

A heuristic solution to this problem is to investigate the filtering kernel as it moves over each pixel neighborhood, and select and mask some of the pixels that does not belong the object. A threshold can be set that will eliminate far away points in terms of depth values. Assuming the depth values are in millimeters (as in our case with Kinect), in our experiments a value 50 (5 centimeters) worked fine. Equation(3.5) shows the modified masked filter. W define the filtering kernel, D is the depth map and M is the mask. M can also be calculated by object map which has been calculated previously in background removal.

$$D(p,q) = \frac{1}{\sum M} \sum_{q \in W} D(q).M(q) \tag{3.5}$$

After applying the masking scheme described we get an output similar to Figure 3.5(b). We see that artifacts in the boundaries do not appear in the output point cloud. It is also seen that the output point cloud after mean filtering is smoother and noise level is decreased.

Figure 3.5: Raw(left) and Mean filtered(right) point cloud with masking. Bottom row is a closer look onto head part.



Source: This figure has been prepared by Bahtiyar Kaba**.**

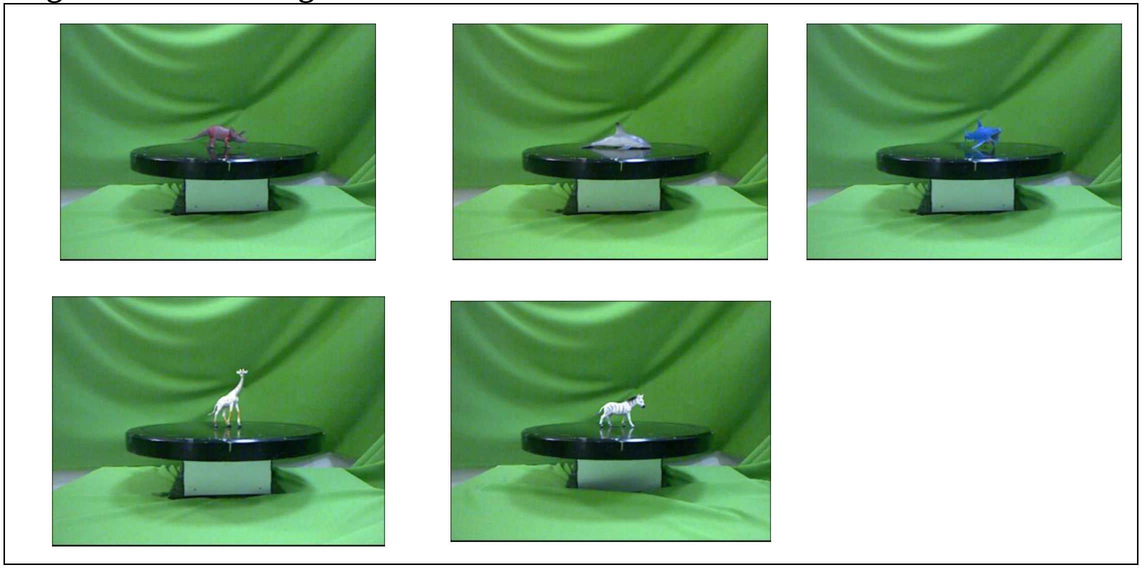
To quantitatively analyze the filtering, we calculate its accuracy value as described in section 3.2 . Accuracy is normally calculated on reconstructed and ground truth complete models, but for assessment here, we will use it on individual point clouds. Our procedure for this analysis is as follows:

a. 30 random depth maps are chosen from the Doumanoglou dataset each from either these 5 objects: rhino, dolphin, shark, zebra, giraffe. (Figure 3.6)
b. The depths map are filtered with the respective algorithm, background removal is performed.
c. Resultant images are reprojected to 3D, and point clouds are extracted for each depth image.
d. Each point cloud is registered to its corresponding ground truth mesh with ICP algorithm. (i.e, a point cloud from rhino object is registered to ground truth rhino)
e. Accuracy values are calculated for each point cloud.

In Figure 3.7, accuracy results comparison between the raw depth maps outputs and mean filters are compared. The mean filter window size is given as 3, and accuracy results are calculated for per cent thresholds between 75-100.

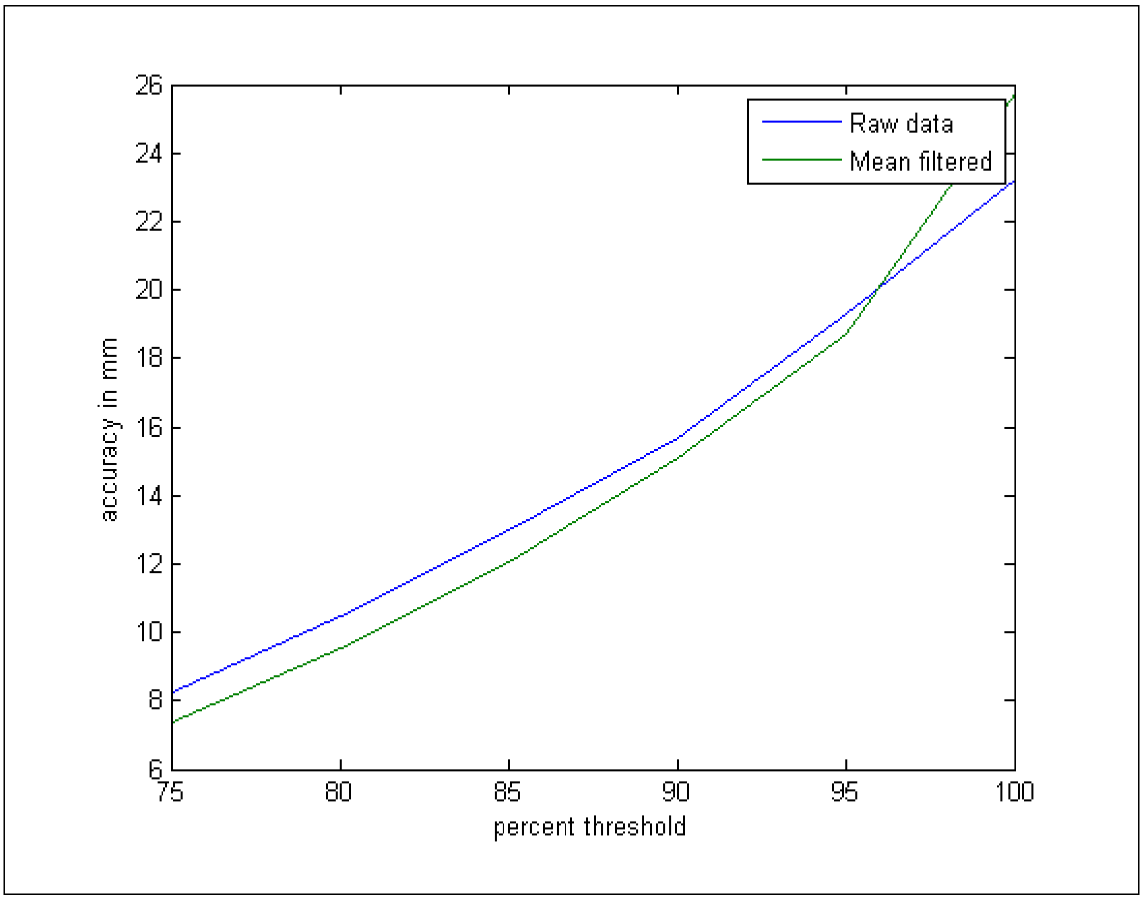
The results as seen in Figure 3.7 which clues that mean filtering improves the output slightly, however, as other parts of 3D scanning pipeline process the data, the output results may differ for the better or worse. Therefore, we made an analysis after alignment and surface reconstruction which will be discussed later. A threshold value of 100 per cent is another way of saying the distance of point who is furthest, and a single outlier can drive this value very high. So, we can forgive the unwanted peak of the graph at the end.

Figure 3.6 Doumanoglou dataset



Source: Doumanoglou et al (2013). *A dataset of Kinect Based 3D scans*

Figure 3.7: Raw and mean filtering accuracy comparison



Source: This figure has been prepared by Bahtiyar Kaba.

3.4.4.2 Bilateral filtering

Bilateral filtering does not only focus on spatial closeness but also takes into account the range (similarity) of the pixels (Tomasi and Manduchi 1998). Since it penalizes dissimilarity, the boundary remains crisp, and textured parts are smoothed. This property makes it suitable for depth filtering.

The bilateral filter response is calculated by a combination of weighted neighbors by geometric and spatial similarity. In equation (3.6) , a mathematical description is given. In the equation f is the intensity (depth) value of the neighbour pixel (m,n), and function w is the aforementioned weighting function. The calculation of the weighting function consists of two stages, one for geometric and one for depth similarity. Equation (3.7) is the domain weighting function parametized by σ_d. σ_d can be thought of as the domain of pixels that will take part in the filter response calculation. For example, when σ_d = 3 pixels, function d will have values getting smaller exponentially for pixels that are more than 3 pixel away, their contribution in the weighting function w will be minimal, and for $\sigma_d = 1$, the bilateral filter will only be weighted by depth similarity function . In equation (3.8) , the function r is the depth similarity function which penalizes the output according to our similarity metric. For our case, the l_2 norm of the intensity difference is adequate. Again, there is a parametrization variable σ_r which we can use to adjust how the filter will react. Here, giving this variable a value of 1 will result in the bilateral filter as gaussian filter since the intensity weights will not make any effects on the weights, but only the geometric distance.

The weighting function w is finally calculated by multiplying the geometric and depth similarity functions. The neighboring pixels (the window) of the kernels is defined by (m,n) given as parameters to the functions. Although, the domain function already penalizes far away pixels, for implementation purposes we use one more parameter, window width, which will define the window on which the calculations are to be performed. (m,n) parameters will thus have values in the range [-width, +width]. This would also spare some computation costs, especially when σ_d has smaller values.

$$D(p,q) = \frac{\sum_{m,n} f(m,n)\, w(p,q,m,n)}{\sum_{m,n} w(p,q,m,n)} \tag{3.6}$$

$$d(p,q,m,n) = e^{-\frac{(p-m)^2 + (q-n)^2}{2\sigma_d^2}} \tag{3.7}$$

$$r(p,q,m,n) = e^{-\frac{\|f(p,q) - f(m,n)\|^2}{\sigma_r^2}} \tag{3.8}$$

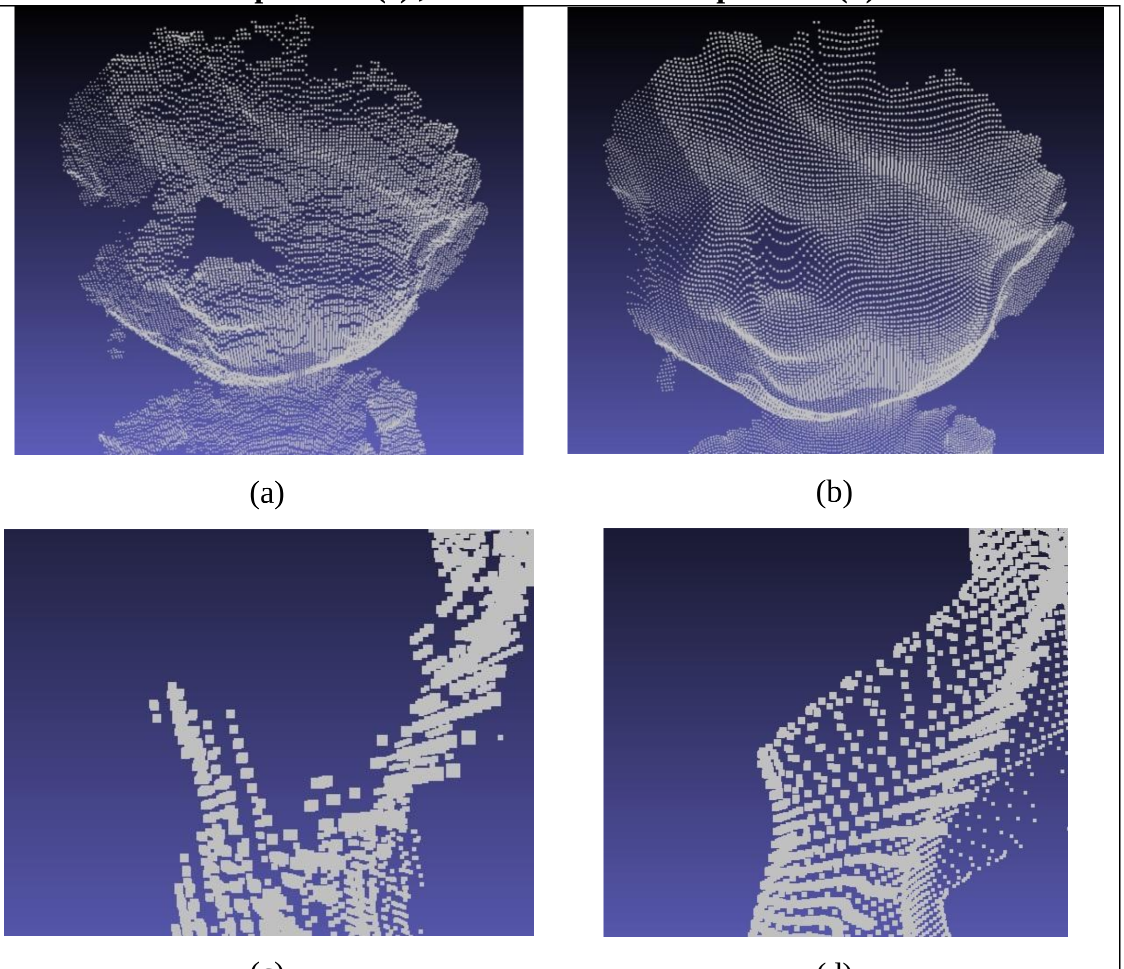
$$w(p,q,m,n) = d(p,q,m,n) * r(p,q,m,n) \tag{3.9}$$

Investigating the mathematical definition of bilateral filtering, it is obvious that this type of filtering is appropriate to depth filtering without much modification. Given proper parameters, the weighting function will eliminate out-of-object pixels since their depth values will be significantly different. Moreover, incorporating depth similarity in the filtering process is expected to preserve the flow of geometry in the object.

An example output is shown in Figure 3.8 . The cloud reprojected from the filtered depth map is smoother, and the boundary pixels are not distorted. However, it can also be noted that some small details are gone, too. To give an example, in the raw cloud, the nose part of the boy object is more pointy (similar to the real object), but in the filtered cloud the nose is smoothed and the pointy end is not prominent. In Figure 3.8 (c) and (d) , a visual comparison can be seen. When we are making our quantitative analysis, we will investigate what this problem can become.

Similar to our previous quantitative analysis on Gaussian filtering, we performed tests on the Doumanoglou dataset with bilateral filtering. The 30 depth frames are processed with bilateral filter and the parameters are given as width = 5, σ_d=3, σ_r= 0.01 . In Figure 3.9, a graphical representation our output is shown. The raw and mean filter outputs are same with the previous comparison. Now, we see some slight improvement in bilateral filter output compared to mean filters, and more improvement compared to the raw depth map. The higher threshold values (above 95 per cent) still continues to be a problem though. But as we discussed earlier, they can be neglected for our analysis.

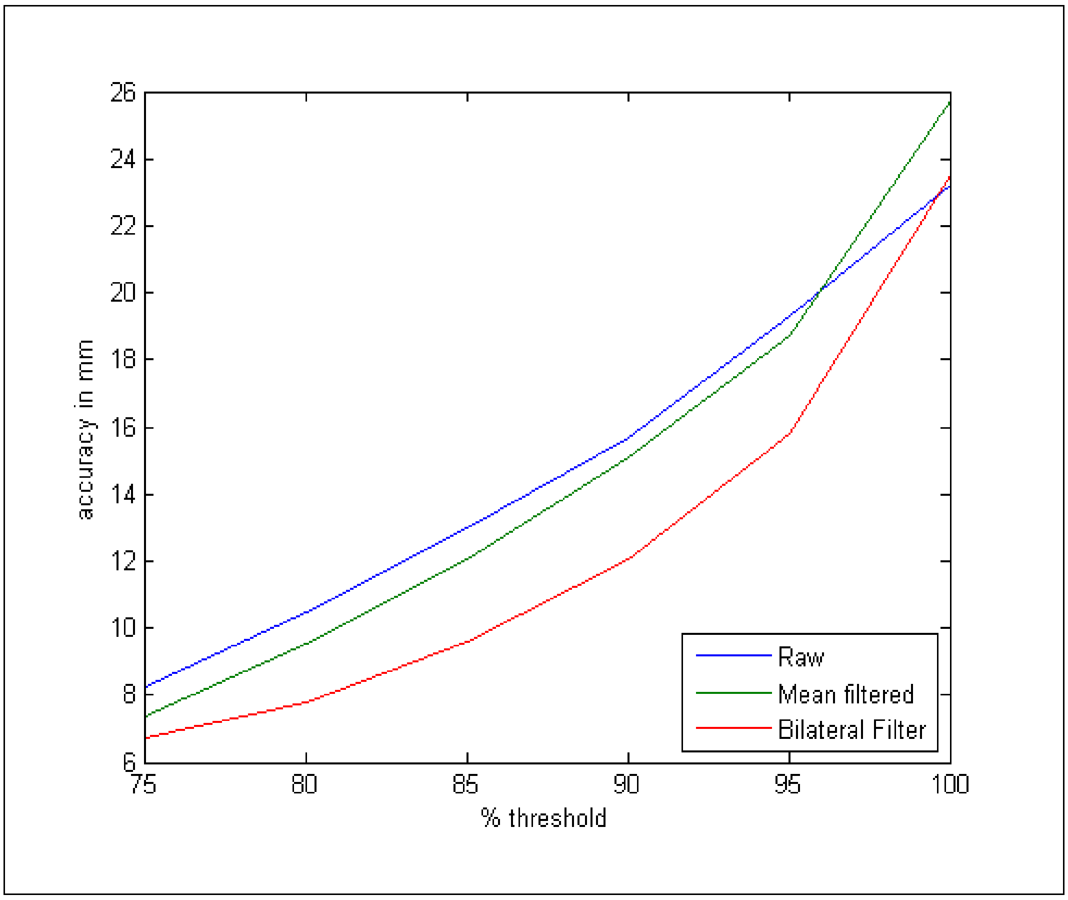
Figure 3.8: Point cloud from raw depth map(a) and bilateral filtering (b) , raw cloud close up of nose(c) , filtered cloud close up of nose(d)



Source: This figure has been prepared by Bahtiyar Kaba.

These results suggests that bilateral filtering can be a good method of choice for our 3D scanning system. However, these accuracy values are calculated without the latter processing steps. They might affect the final result, and these effects can be dependent on the filtering method applied. We will look into the results of these later when we discussed the relevant alignment and surface reconstruction algorithms.

Figure 3.9: Bilateral filtering accuracy comparison



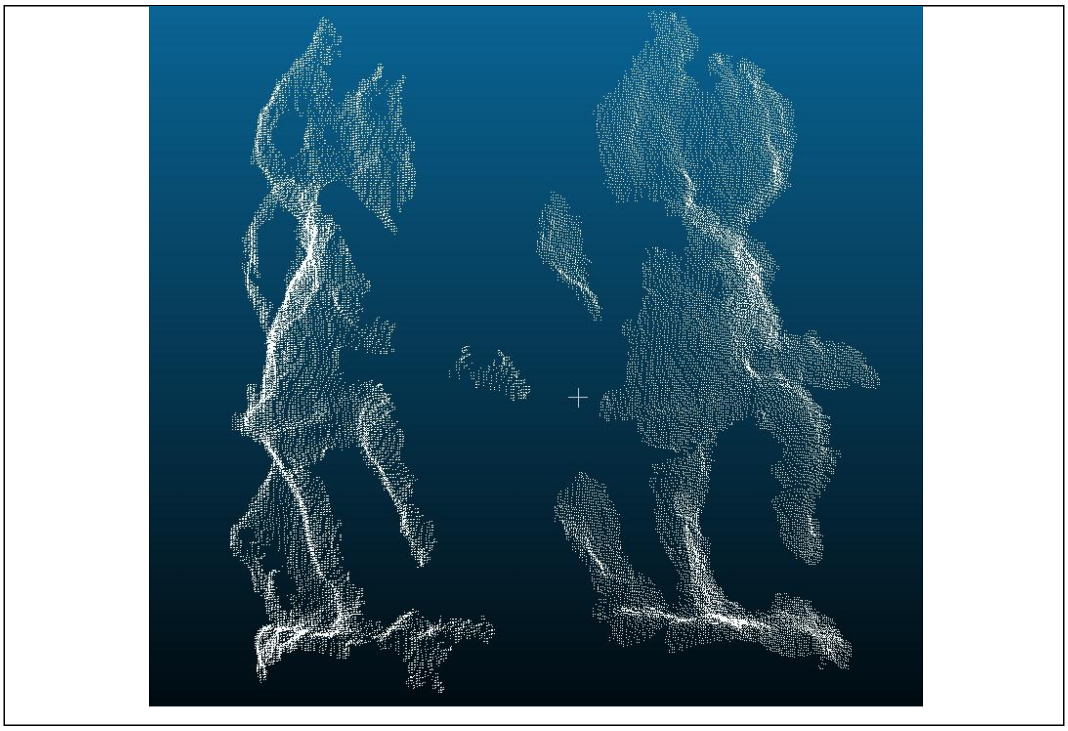
Source: This figure has been prepared by Bahtiyar Kaba.

3.5 SCAN ALIGNMENT

3.5.1 Coarse Alignment

After the depth maps are filtered and back-projected to point clouds, the point clouds which are on different coordinate systems will be aligned. An example of the same object scanned from 50 degree intervals is seen in Figure 3.10.

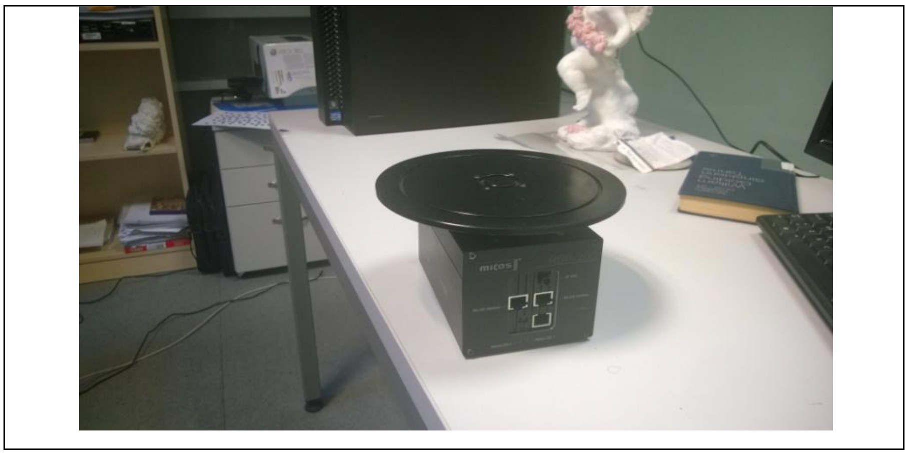
Figure 3.10: Same object scanned from 50 degree interval



We proposed a controlled motion to our system, therefore, we have the coarse information about the alignment already. Applying a 50 degree rotation to the second cloud in Figure 3.10will coarsely align it with the first one.

We place the objects on top of rotating stage which can given commands to spin around specific angles. We use Micos Pollux RSP 200 (Micos website[5]) to realize the controlled rotation motion in our system. See Figure 3.11.

Figure 3.11: Micos Pollux RSP 200 Rotating Stage



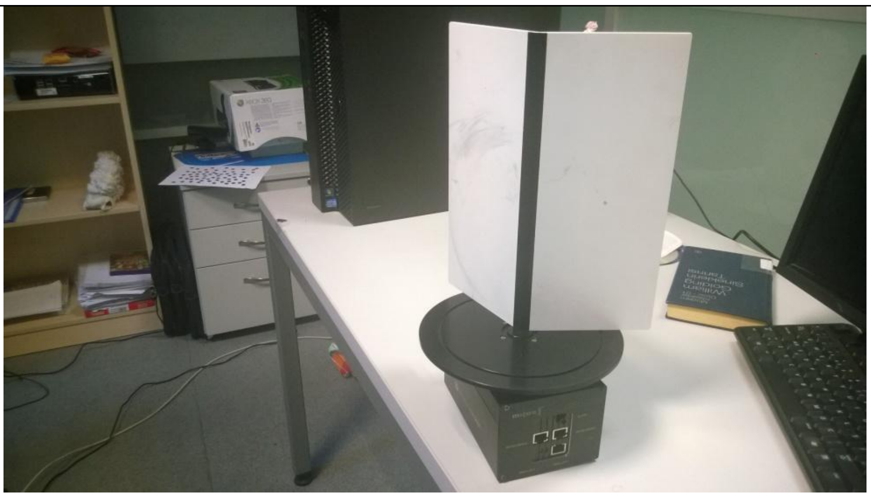
Source: This figure has been prepared by Bahtiyar Kaba.

[5] Micos website : http://www.pimicos.com/web2/data/discontinued/1,5,150,rsp200.html

For finding the transformation applied to the clouds, it is necessary to know the axis of rotation. The specified degrees of rotation occurs around the axis that passes through the center of rotating stage.

To find the rotating axis, we incorporate a calibration tool which consists of two intersecting planes as in Figure 3.12. When we take a scan of the calibration tool by placing it on top of the rotating stage so as to align its intersection line with the central axis of the stage, the output point cloud will have two prominent planes provided that the background is removed.

Figure 3.12: Calibration tool



Source: This figure has been prepared by Bahtiyar Kaba.

We can estimate the parameters of these planes using RANSAC algorithm (Fischler and Bolles 1981, Xu and Lu 2012). With RANSAC algorithm, it is possible to get the coefficients of the two planes (rows of multiplier matrix in equation (3.10)) The intersection line of these planes is defined by solving the plane equations. The solution to the linear equation in (3.10) is an infinite set of points if it exists which defines the line. Equation (3.11) is the representation of line such that (x_1, y_1, z_1) and (x_2, y_2, z_2) are any different points from the solution set and t is any real number. The rotating axis is found this way.

$$\begin{bmatrix} a_1 b_1 c_1 d_1 \\ a_2 b_2 c_2 d_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0 \tag{3.10}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + t \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \tag{3.11}$$

After the rotation is applied, the clouds are coarsely aligned and ready for the fine alignment stage.

3.5.2 ICP

Iterative Closest Points (ICP) refers to a class of algorithms that finds the rigid-body transformation between a pair of clouds by minimizing an error between the input clouds (Besl and McKay 1992).

The ICP algorithm works as follows: First, corresponding points between the pair of clouds are found. The corresponding points can be found by their curvature or color similarity in the neighborhood. After the corresponding points are found, an initial error is calculated based on a specific metric. Several metrics have been proposed such as point-to-point distance and point-to-plane distance. The algorithm starts to work iteratively, and at each iteration the objective function (the RMS of the error metric between the corresponding points) is recalculated and the necessary transformation to minimize its value is estimated. At next iteration, the cloud is transformed with the estimated matrix and the same procedure is reapplied.

As the iterations continues, the fit error between the two clouds will converge. When the objective function converges the algorithms stops and outputs the multiplication of all inter-iteration transformations.

In our 3D scanning setup, we need to scan the object from many view angles. Convinced that ICP is a good algorithm for aligning a pair of overlapping point clouds, we can chain apply it to each subsequent point cloud. A mathematical description of what we are trying to achieve is shown in Equation (3.12) . T_i denotes the transformation estimated by ICP between cloud (i) and cloud (i-1) . The proper alignment of nth cloud is done by transforming it to the (n-1)th cloud which have been transformed to (n-2)th and it goes likes till the first cloud.

This method is good if the point clouds to be registered are not covering the full object.

$$T_1 = I \qquad\qquad n=1,2,... \tag{3.12}$$

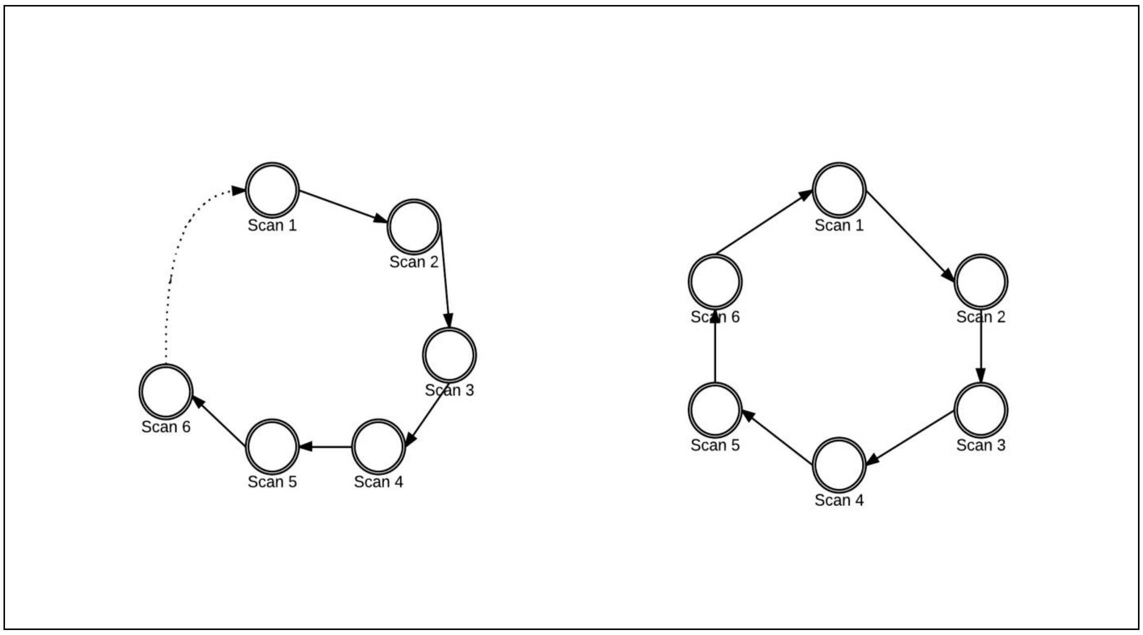
$$P_n' = \left(\prod_{i=2}^{n} T_i\right) P_n$$

When the coverage of the scans is to capture the full object, the accumulation error poses a problem. When each subsequent frame is registered with ICP, there is a minimal error in between them. When the nth cloud is to be aligned with the first cloud(reference), the transformation matrix is found by multiplying the transformations of all subsequent clouds until itself. Therefore, the error between the reference cloud(first) and the nth cloud becomes larger. And, this problem increases towards the last cloud in the set, which should supposedly be very close to the reference cloud. This problem is known as the loop-closure problem.

Figure 3.13 depicts a simulation diagram of the loop-closure. Supposedly, 6 scans around an object are captured with equal rotations. After the alignment process is done, we expect the poses of the scans to be as in right of Figure 3.13 . However, when the error is minimized between each subsequent frame, the poses of the scans look as in left of the figure. The error between scan 6 and scan 1 is very significant. This is due to the fact that alignment is optimized locally between subsequent scans without taking into account the global positioning.

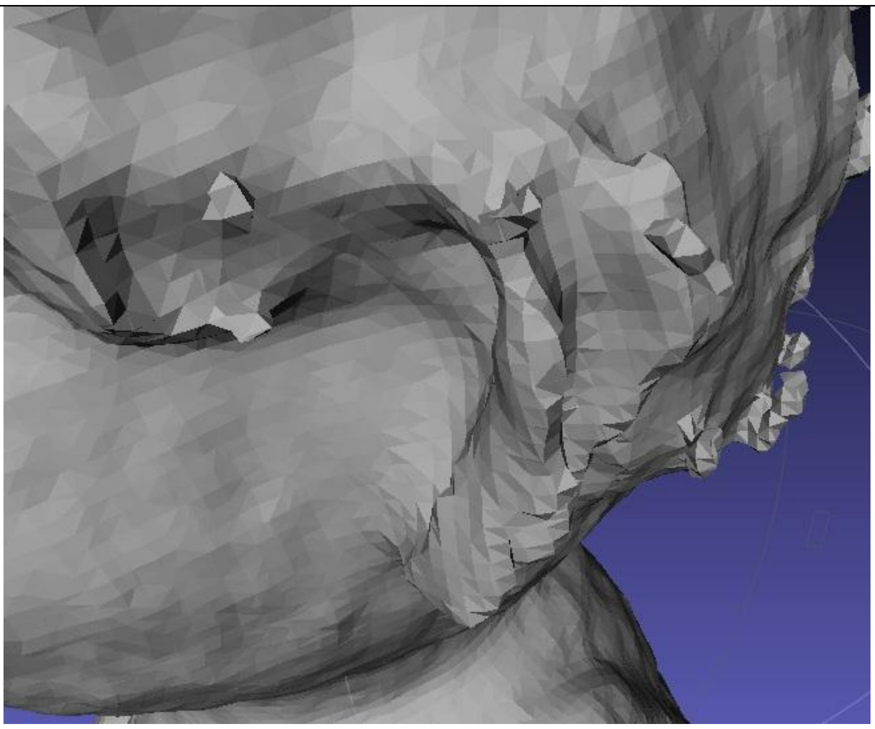
A practical result of this shown in Figure 3.14 . For visualization purposes the point cloud is meshed with Poisson algorithm. The figure zooms into the ear part of boy object. Since the loop closure has not been done properly, and the first (reference cloud) pose is significantly different than that of the last cloud, the ears (not only ears, the clouds as a whole) does not overlap.

Figure 3.13: Loop closure problem



Source: This figure has been prepared by Bahtiyar Kaba.

Figure 3.14: Erroneous output mesh due to loop closure



Source: This figure has been prepared by Bahtiyar Kaba.

3.5.3 Graph Based Global Alignment

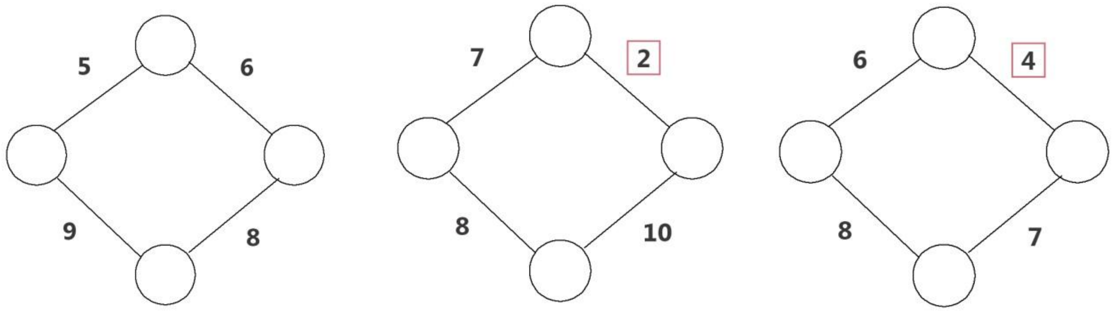
The loop closure problem stems from the fact errors are minimized locally leaving a large error between the last and first cloud that are supposed the close the loop. The solution, therefore, starts by suggesting to distribute this large error to all transformation evenly. An even distribution will prevent the error to accumulate in one place, and avoid the loop closure problem. One can argue that distributing the error will lead other transformations to be distorted; however, when the error is shared between all of them, its contribution will be negligible.

Lu and Milios (1997) suggests a graph optimization approach to make this even distribution of error. The vertices of the graph are point clouds and the global transformation matrices regarding to a common world coordinate system. The edges of the graph represent the constraints between the vertices(point clouds) and have the correspondence data between them. The weight of an edge is measured by the distance of the correspondences for a particular metric.

The algorithm uses ICP for pair wise error minimization with constraints defined by the global pose. These constraints are adjusted to keep edge weights between vertices globally even, thus preventing error accumulation on any of them.

Figure 3.15 depicts this constrained error minimization in an example graph diagram. At the start of an iteration, assume the graph the structure is as in (a) and edge having a weight of 6 is being optimized. If the objective function only considers the pair of vertices of this edge, the result might be as in (b). Here, minimizing the error for one edge caused other edges to have higher weights since the global transformation matrices are modified for each of them. But if the minimization is constrained such that it will not go as low as to 2, and it will go 4 instead; the weights of other edges will not be altered too much as shown in (c). Therefore, the objective function of the ICP is constrained such that not to allow this kind of behavior.

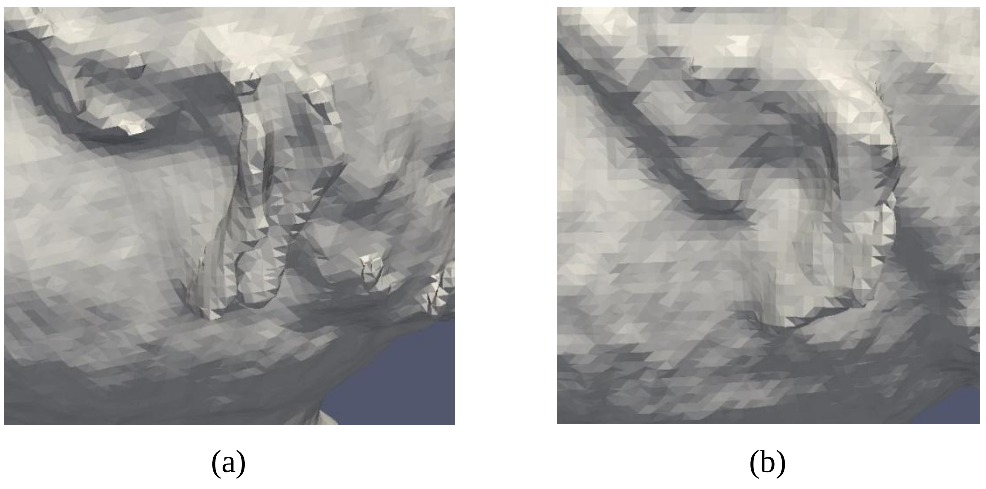
Figure 3.15 Graph constrainted iteration, initial (a) , no constraints (b) , constrainted objective (c)



Source: This figure has been prepared by Bahtiyar Kaba.

Previously shown erroneous mesh mitigated by global based alignment is shown in Figure 3.16 .

Figure 3.16 Globally optimal alignment: loop closure (a), globally optimal alignment(b)

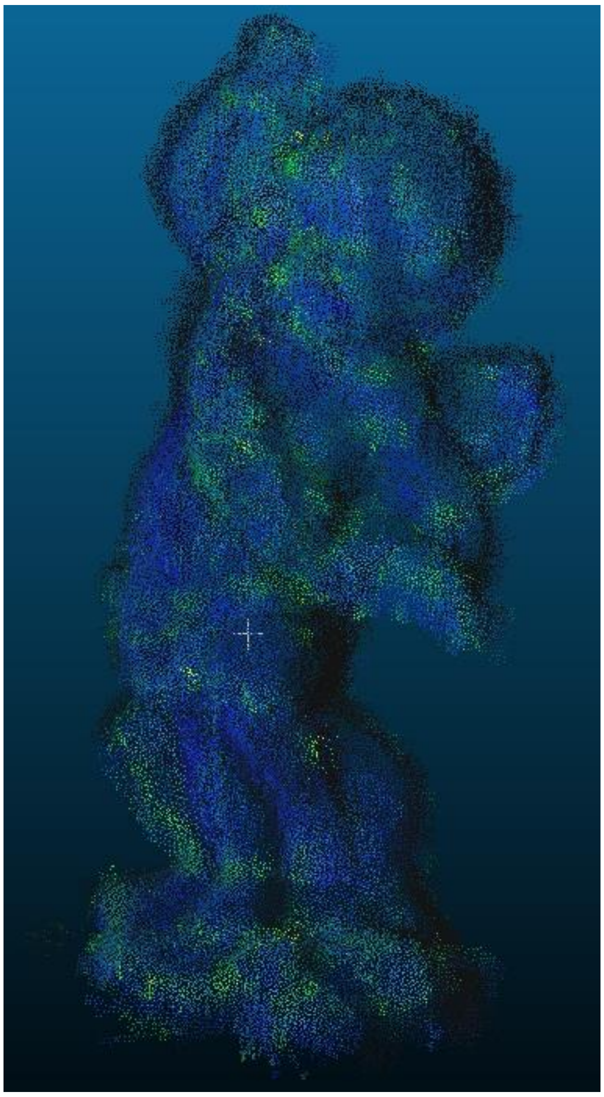


3.6 SURFACE RECONSTRUCTION

3.6.1 Poisson

Scanned clouds from different view angles around the object have been transformed to be aligned in a common reference(global) coordinate system and unified to produce an integrated cloud. The vertices of this cloud define a digitized representation of the real object sampled at discrete point. Figure 3.17 shows an example point cloud registered and concatenated from many point clouds . Depending on the application, the scanning process can be reported to have finished here.

Figure 3.17: Registered and concatenated point cloud



Source: This figure has been prepared by Bahtiyar Kaba.

In this work, we aim to output polygonal mesh models which will contain the edge information together with the vertex data. To do this, the final (registered, unified) cloud needs to be processed to find the faces of the polygons.

Poisson reconstruction algorithm (Kazhdan et al. 2006) is a surface triangulating algorithm from oriented point sets (i.e. with normals). It uses the implicit function approach to define and solve the surface equation, and outputs a polyhedral mesh representation (as triangulated faces).

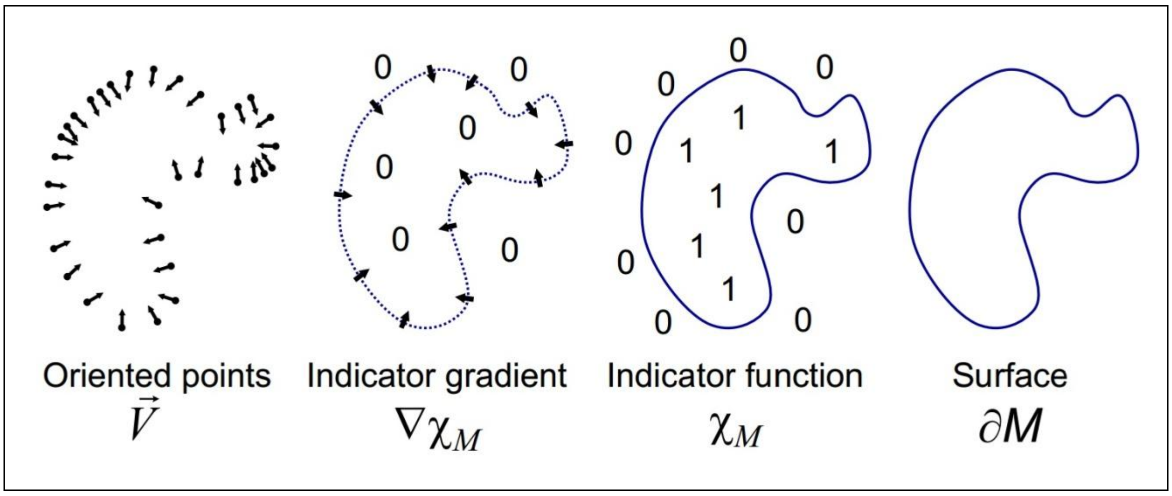
A 3D indicator function χ, defined as 1 inside the surface model and 0 outside, is used to extract an iso-surface which will approximate the surface of the actual object. Now, it is necessary to find this function. The gradient of the indicator function χ is constant for the most cases except along the surface. Therefore, the normals of the point set are like the samples of the gradient field of χ . Finding the function is possible by finding the function χ whose gradient best approximates the sample vector field (the normals). A 2D example of this scheme is represented in Figure 3.18 and a real world example is shown in Figure 3.19 . As the input to the algorithm a set of oriented points (point cloud with normals) is given, which is then by processed with the given octree depth and produces the triangulated mesh (Figure 3.19).

Equation (3.13) shows this relation mathematically. V denotes the vector field sampled by the input point clouds' normals. The objective is to minimize the distance between the gradient of the indicator and this vector field. However, V is generally not integrable, so some manipulation is necessary. If we consider a random point p, equation (3.13) suggests that the divergence of the gradient of χ should be equal to divergence of V around p. Without loss of generalization, equation (3.14) can be derived. The equation in this form is in the form of a Poisson equation (Winslow 1966).

$$\min_{\chi}\left\|\nabla\chi - \vec{V}\right\| \rightarrow \nabla\chi = \vec{V} \tag{3.13}$$

$$\Delta\chi \equiv \nabla . \nabla\chi = \nabla . \vec{V} \tag{3.14}$$

Figure 3.18: Poisson 2D example

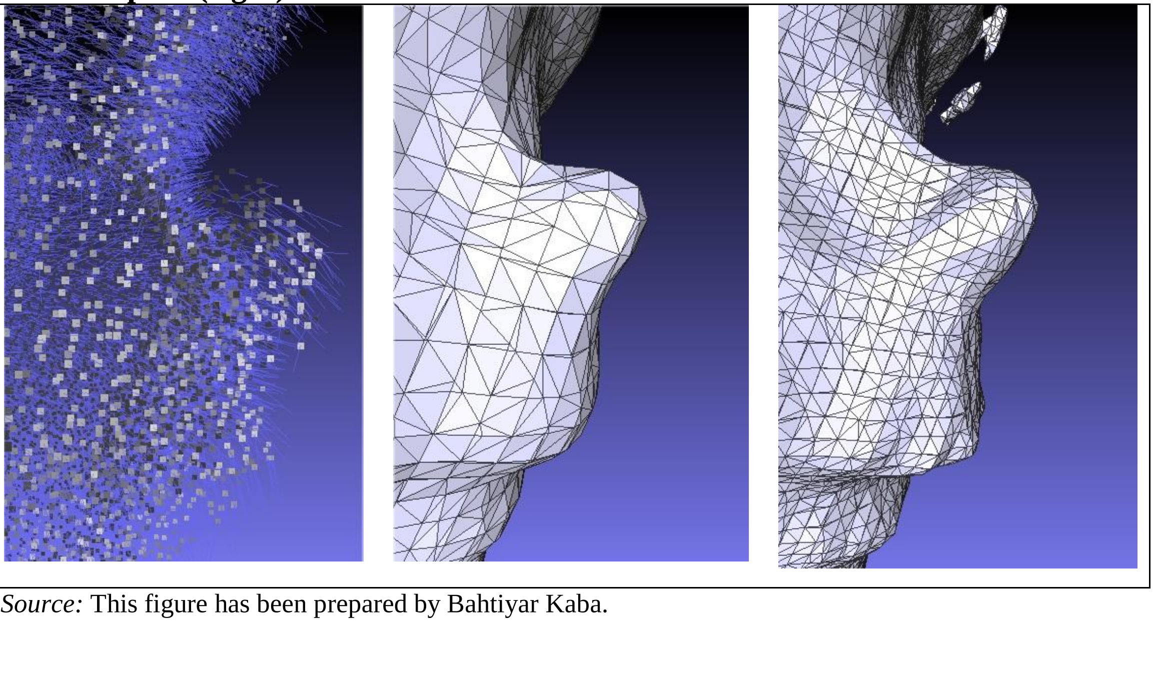


Source: Kazhdan et al., (2006) *Poisson surface reconstruction*

For implementation, the space can be partitioned into a regular grid, and calculations can be done on that quantization level. To capture finer detail, the resolution of the grid can be increased but after some point it will become impractical (memory constraints) . To keep fine detail intact and achieve practical execution, an octree representation is used. This octree will be given a maximum depth parameter. The octree will be generated such that the depth will be small for regions far from the surface (because it is not important to keep lots of emtpy cells) and will approach max depth around the surface.

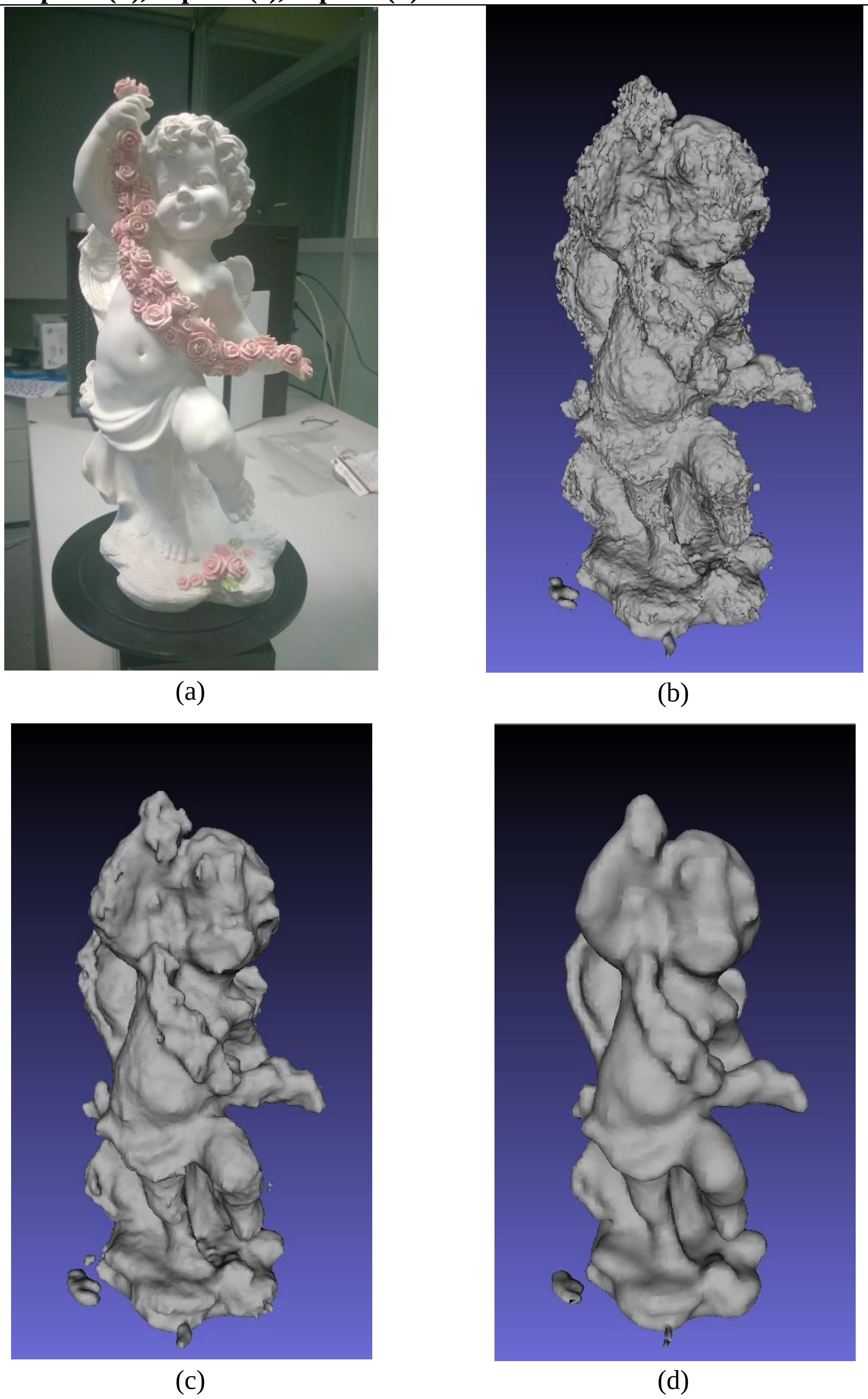
For our experiments, we used the maximum depth parameter for getting and comparing results. The depth parameter will affect the detail of the output mesh. If higher values are given, the grid resolution around the surface will be higher and more detail will be captured. However, as we see in Figure 3.20, increasing detail will lead a lot of noise to contaminate the output. In the figure (a) shows the color image object that is scanned, the angle object, (b) is the output reconstructed model with depth set to 9. The resolution of the grid where the Poisson equation is solve is fine-grained. And as expected, the output mesh has too much crease. As the octree depth is lowered to 8 and 7, the creases vanishes and output mesh becomes smoother. The mesh reconstructed with depth 7 is the smoothest and most visually appealing.

Figure 3.19 Oriented point set(left) and application of Poisson depth 7(middle), depth 8(right)



Source: This figure has been prepared by Bahtiyar Kaba.

Figure 3.20: Poisson algorithm with different max octree depths color image(a), depth 9 (b), depth 8 (c), depth 7 (d)



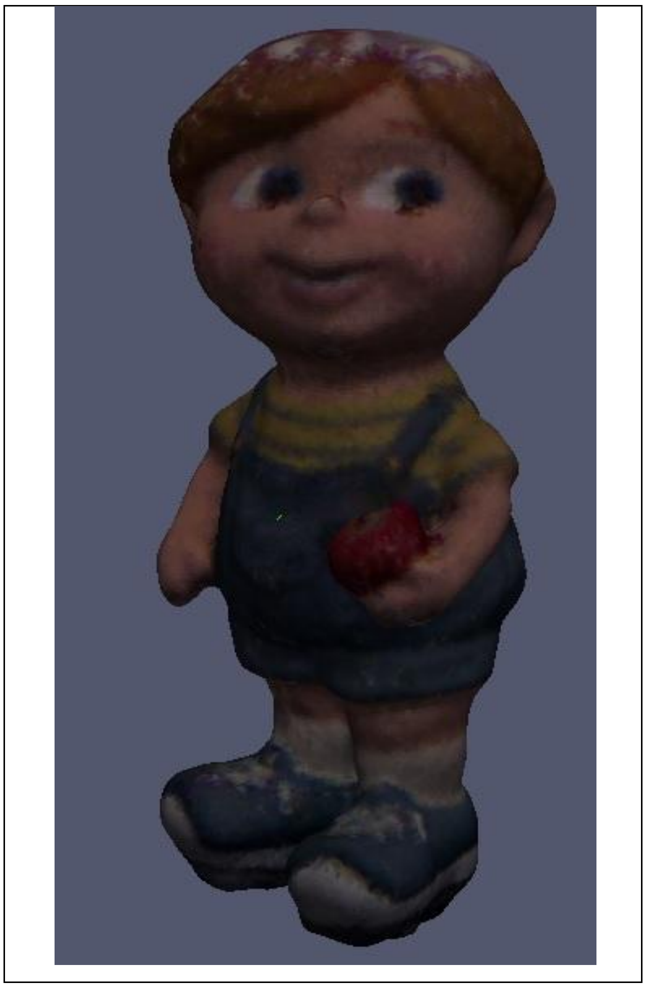
Source: This figure has been prepared by Bahtiyar Kaba.

The smoothing of the surface due to lowering octree depth poses another problem. While the noisy parts are removed, the high frequency detail that exists in the actual object might have been removed, too. Again in Figure 3.20, notice how the pits of the eyes of angel start to be less distinctive as the depth value is decreased. For another example, the navel (belly button) which can slightly observed in depth 9 and depth 8 has completely disappeared at depth 7.

3.6.2 Coloring

Although it is not one of our primary objectives, we have tested a basic coloring scheme to our 3D reconstruction scheme. Kinect has an RGB sensor besides the depth sesnsor, and calibration parameters between these is known. Therefore, when the point clouds are captured, color information for each vertex is stored, too. When the triangulated mesh is built, vertices of the mesh is given the color of the nearest point in the input concatenated cloud. During renderin, the color of face is interpolated between its three vertices. An example output can be seen in Figure 3.21.

Figure 3.21: Color mapped mesh



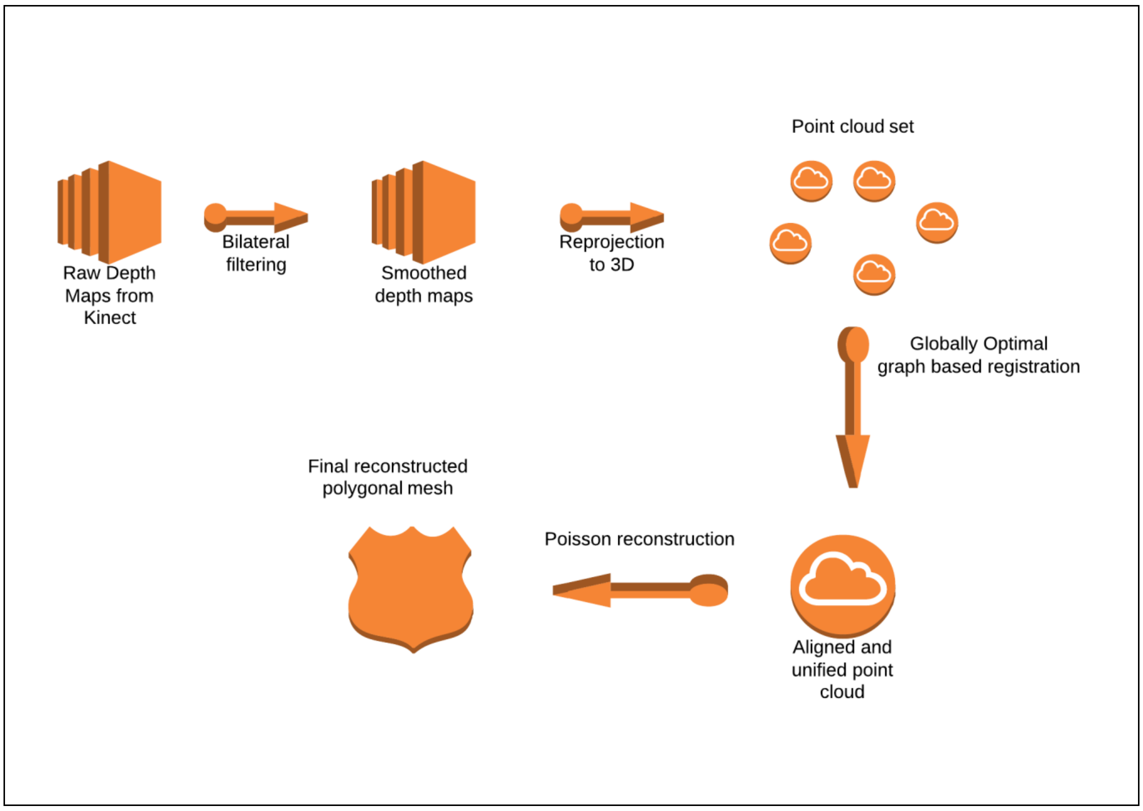
Source: This figure has been prepared by Bahtiyar Kaba.

3.7 3D SCANNER WITH KINECT

With the algorithms we tested out , we describe the our scanner system setup. In Figure 3.22, schematic representation of the system is given. In Appendix B, a pseudo code of the modules is presented. The depth maps are acquired with a Kinect sensor and bilateral filtering applied to them for smoothing and noise reduction. For later processing, normal values are estimated, too . Then, the acquired depth maps are back-projected to 3D coordinates with camera calibration parameters of Kinect, the known rotation transformation are applied for coarse alignment.

The unregistered point clouds is then processed to find correspondence information between them that will be used for registration. Then, the point clouds are placed into the graph based alignment algorithm setting the edge weights according to the estimated correspondences. The alignment module transforms the point clouds onto a common world coordinate system, and outputs a concatenated(unified) point cloud. This unified cloud is ready for surface reconstruction for Poisson. In analysis and evaluation section, the comparisons are made based on the meshes that are reconstructed as described here.

Figure 3.22: System setup schema


Point cloud set
Raw Depth Maps from Kinect
Bilateral filtering
Smoothed depth maps
Reprojection to 3D
Globally Optimal graph based registration
Final reconstructed polygonal mesh
Poisson reconstruction
Aligned and unified point cloud

Source: This figure has been prepared by Bahtiyar Kaba.

4. ANALYSIS AND EVALUATION

In this chapter, we analyze the methods and algorithms that have been discussed previously and used in our system setup.

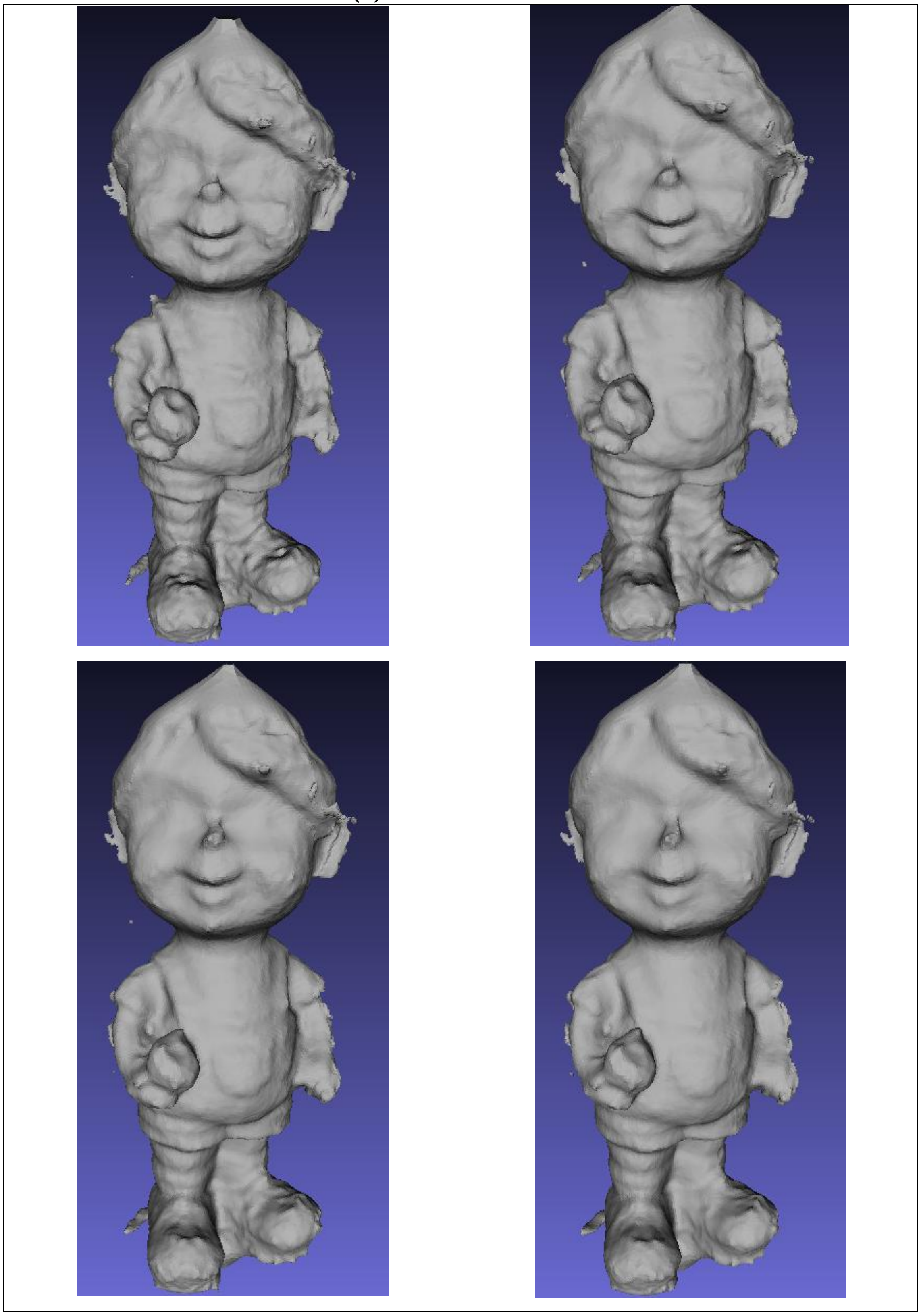
4.1 FILTERING ANALYSIS

On chapter 3, mean and bilateral filtering methods have been discussed and some analysis on them have been presented on the relevant sections. There we have investigated how they perform on single depth maps and compared their accuracy values (refer to the relevant sections for more insight).

Here, we apply the filtering methods on the whole set of scans for the boy object and evaluate their performance.

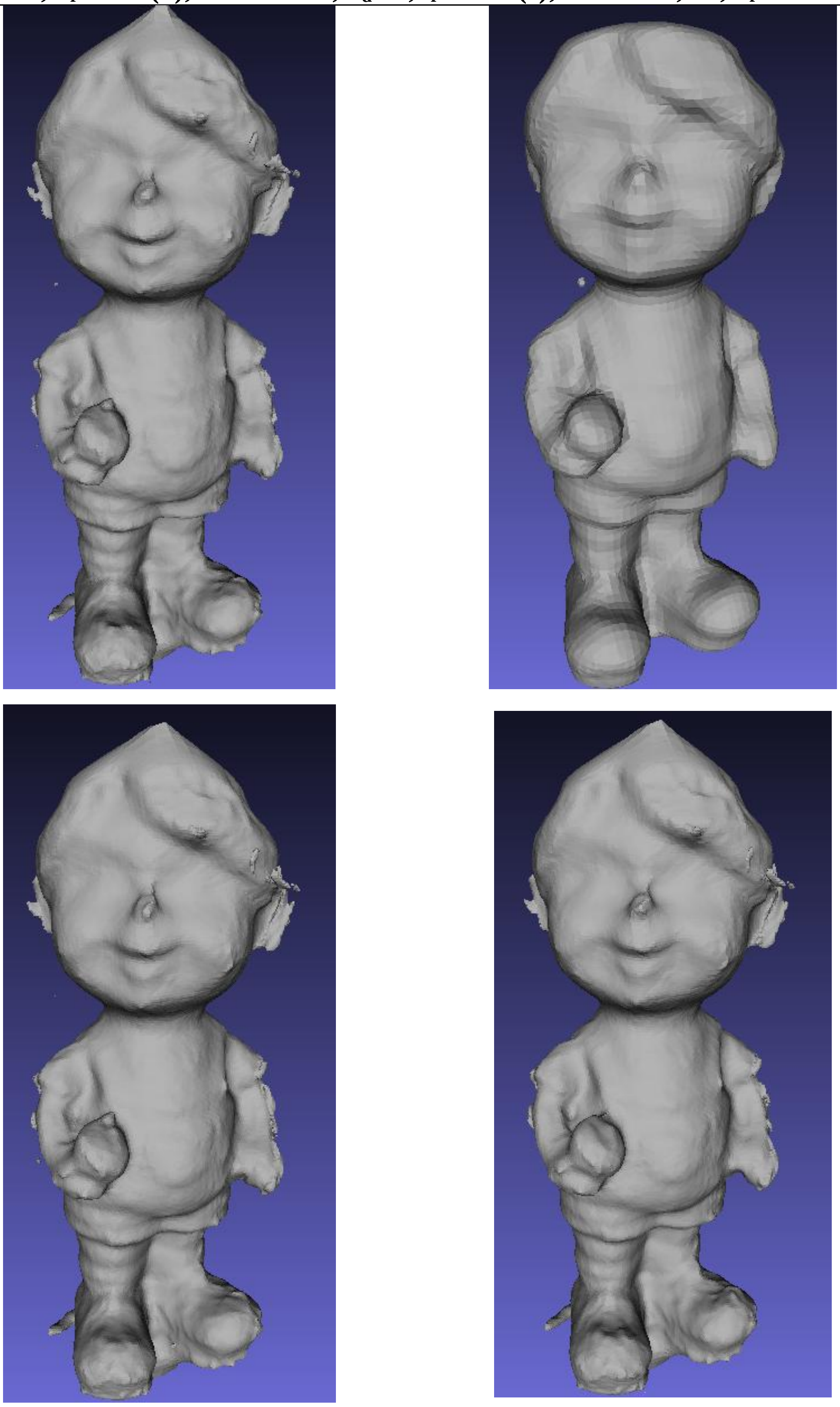
Figure 4.1 show how the effect of mean filtering occurs on the output point clouds. The scanned clouds are filtered with a mean kernel of sizes 3 (b), 5 (c) and 7 (d) . (a) shows the raw point cloud reconstruction for reference. The registered clouds with ICP, are then triangulated with Poisson algorithm with an octree depth of 8 and no post processing is done. As seen in (a) , Poisson algorithm is capable of removing some noise and smoothing the results, and it will be smoother with less detail as discussed in section 3.5 . In (b) , we see that we can remove some unwanted noise by keeping the detail at a certain level and without resorting to a lower octree depth in Poisson. (c) and (d) shows that we can continue to widen the kernel size for further smoothing. The visual inspection suggests that a kernel size of 5 is optimum because noise is removed greatly and as the kernel size gets wider the details start to be lost.

Figure 4.1: Raw (a) , mean filter with width 3 (b), mean filter with width 5 (c) , mean filter with width 7 (d)



Source: This figure has been prepared by Bahtiyar Kaba.

Figure 4.2: Bilateral filtering results. width = 3, σ_d = 3, σ_r= 0.01 (a) ; width=3, σ_d =3 , σ_r =0.03(b); width = 5, σ_d =3,σ_r =0.01 (c); widht=5, =3, σ_r = 0.005(d)



Source: This figure has been prepared by Bahtiyar Kaba.

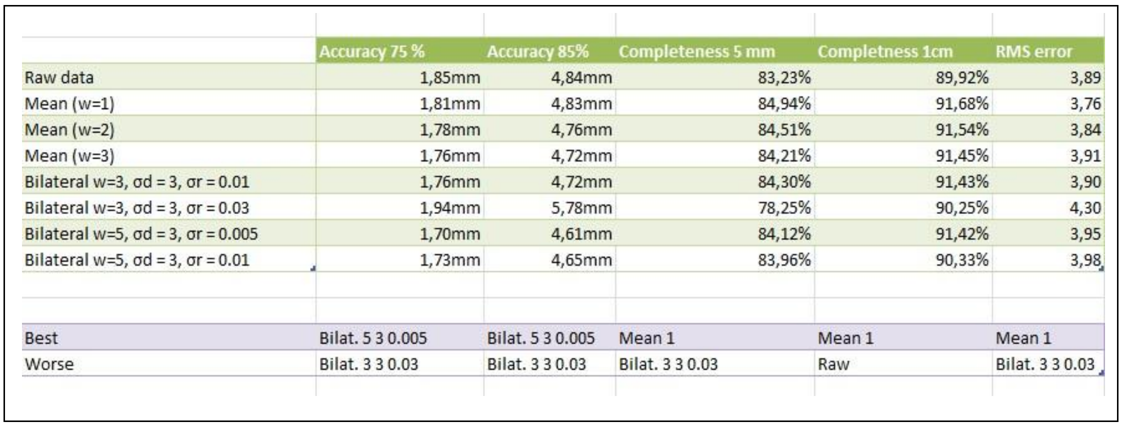
Figure 4.2 shows the reconstruction in the same manner. Some changes in the parameters are reflected in the results. Between (a) and (c) the width of the kernel size is changed. In (a) a lower kernel size produces a more detailed model and also the little noise level difference means we should not sacrifice detail with higher kernel size. Between (c) and (d), the depth range parameter is reduced to 0.005 from 0.01. The effect of this change is not very prominent but when looked carefully we see that reconstructed model (d) has slightly more distinctive creases. The actual difference occurs when the depth range parameter is increased to 0.03 from 0.01 between (a) and (b). Increasing the depth range parameter to 0.03 almost removed all the noise in exchange of fine detail. Notice how in (b) the eyes, the apple in the hand, the shoes are less realistic than that of (a).

A quantitative analysis on performed on the filters with given parameters. In our previous analysis (discussed in 3.3.3.1 and 3.3.3.2) , we concluded that accuracy results are improved when they have been applied on individual depth frames. Here, we make a similar quantitative analysis and calculate the accuracy, completeness and RMS errors of the output of the meshes. For these calculations the vertices of the mesh are used. Table 4.1 summarizes the results.

In Table 4.1, green highlighted area presents the results of the filtering methods with different parameters whose mesh screenshots are presented previously in this section. Accuracy values for 75 and 85 per cents, completeness values in 5mm and 1cm (refer to section 3.2 for description) , and the total RMS error are calculated for each mesh output of the corresponding filter indicated in row headers.

When we inspect the results we see that the quantitative measurements does not differ as much as the visual output changes. Also, the results of filtering does not make a substantial improvement compared to the raw output.

Table 4.1: Quantitative analysis of filtering methods



	Accuracy 75 %	Accuracy 85%	Completeness 5 mm	Completness 1cm	RMS error
Raw data	1,85mm	4,84mm	83,23%	89,92%	3,89
Mean (w=1)	1,81mm	4,83mm	84,94%	91,68%	3,76
Mean (w=2)	1,78mm	4,76mm	84,51%	91,54%	3,84
Mean (w=3)	1,76mm	4,72mm	84,21%	91,45%	3,91
Bilateral w=3, σd = 3, σr = 0.01	1,76mm	4,72mm	84,30%	91,43%	3,90
Bilateral w=3, σd = 3, σr = 0.03	1,94mm	5,78mm	78,25%	90,25%	4,30
Bilateral w=5, σd = 3, σr = 0.005	1,70mm	4,61mm	84,12%	91,42%	3,95
Bilateral w=5, σd = 3, σr = 0.01	1,73mm	4,65mm	83,96%	90,33%	3,98
Best	Bilat. 5 3 0.005	Bilat. 5 3 0.005	Mean 1	Mean 1	Mean 1
Worse	Bilat. 3 3 0.03	Bilat. 3 3 0.03	Bilat. 3 3 0.03	Raw	Bilat. 3 3 0.03

Source: This table has been prepared by Bahtiyar Kaba.

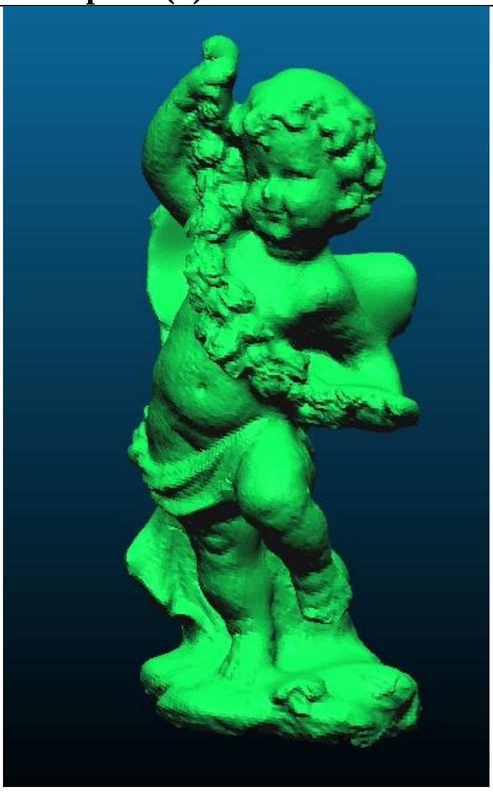
4.2 POISSON ANALYSIS

The implementation of Poisson algorithm uses an octree whose maximum depth parameter can be set. This maximum depth parameter is the allowed octree depth near the surface of the reconstruction. Here, we will analyze how this depth parameter effects the output.

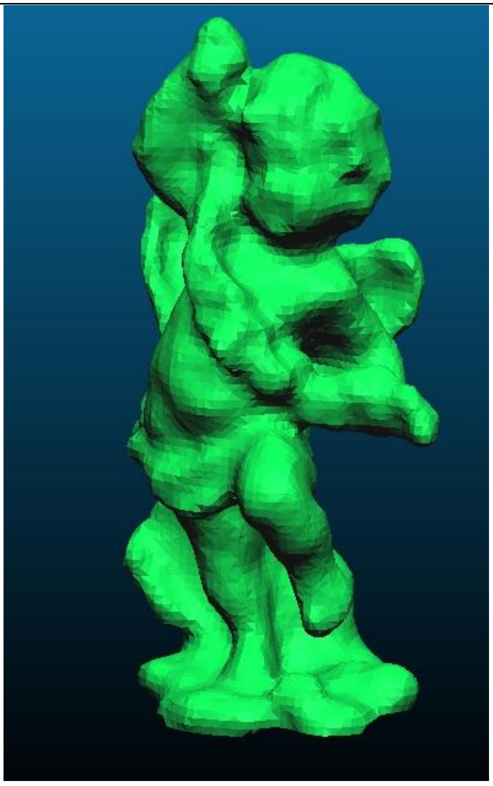
We use our scanning system as described in section 3.6. In Figure 4.3, the application of Poisson filter with different depth parameters are shown. The increasing depth parameter decrease the smoothness of the output final model. For the reconstruction of this model, a depth parameter of 7 or 8 is good depending on the application, but 9 is too noisy. In Figure 4.3(a) , the laser scanned model with Konica Minolta VI 900 is shown. Now using this ground truth data, we make some error evaluation to compare how changing depth parameters performs.

Figure 4.4 shows the color coded error for each of these models. For these colors to make sense, refer to Figure 4.5 which shows the histogram of the error map of reconstructed model with depth 7. The x-axis of the histogram shows the error difference in millimeters. A light green color corresponds to 0, yellow (1-3mm) , orange(4-8mm) and red (> 8 mm). The darker green and blue correspond to negative values (if the point is under the surface of ground truth). Inspecting Figure 4.4, it can be seen that mesh with depth 7 has less red points, and the model with depth 9 has more red distributed over it. This supports that a mesh reconstructed with depth 9 is not satisfactory. Between (a) and (b) , there is not too much difference but the regions that

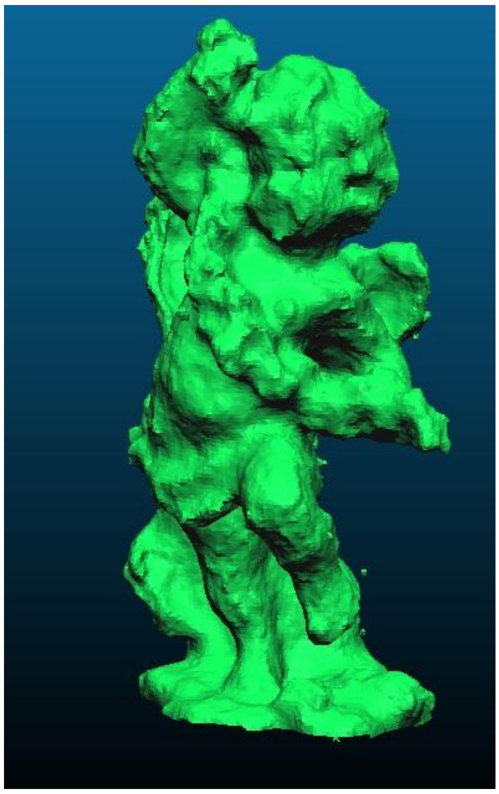
Figure 4.3: Poisson with different depths, ground truth (a), depth 7 (b), depth 8 (c), depth 9 (d)



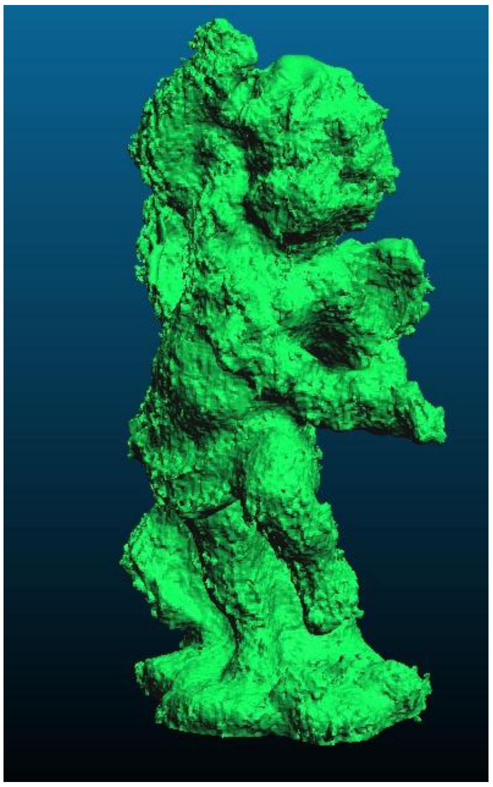
(a)



(b)



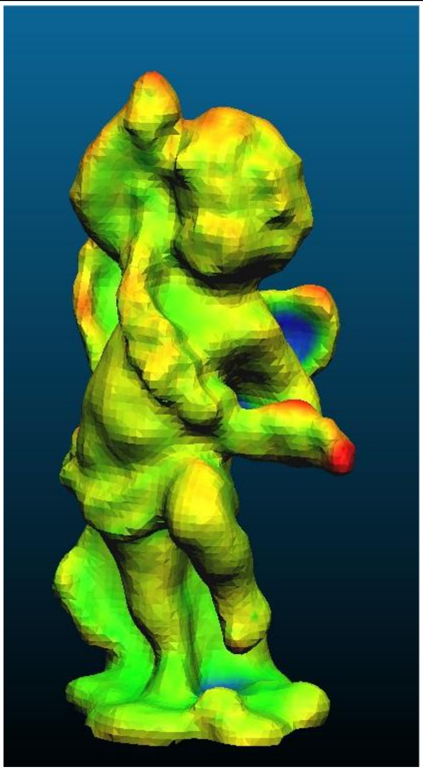
(c)

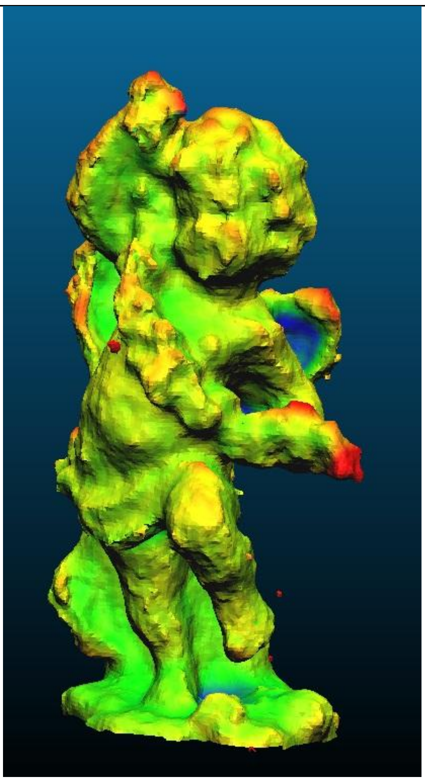


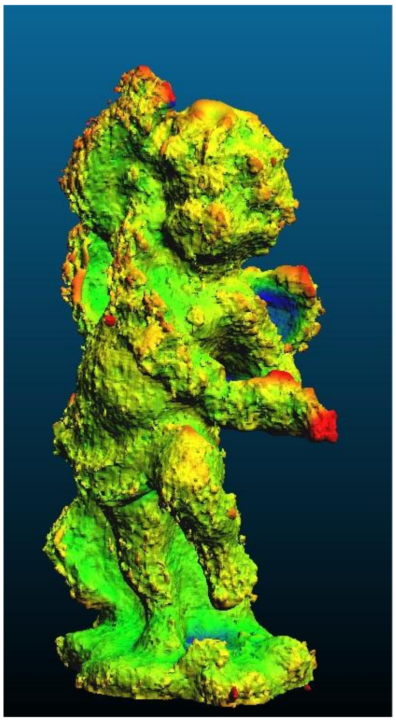
(d)

Source: This figure has been prepared by Bahtiyar Kaba.

Figure 4.4: Color coded error maps for Poisson, depth 7 (a), depth 8 (b), depth (9)

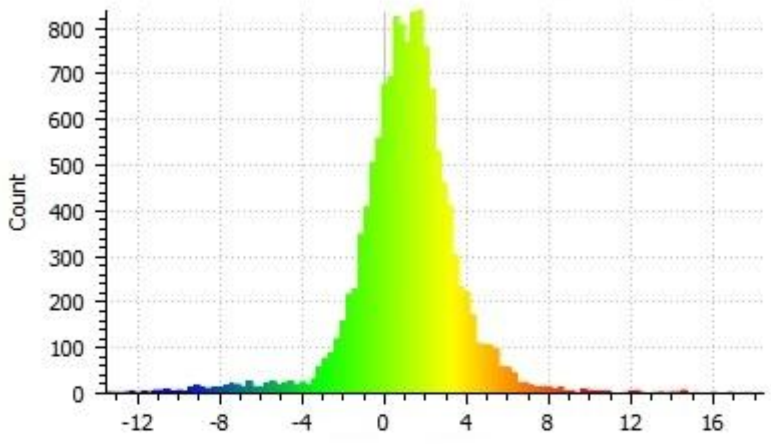

(a)


(b)


(c)

Source: This figure has been prepared by Bahtiyar Kaba.

Figure 4.5: Histogram of the error map, Angel object, Poisson with depth = 7



Source: This figure has been prepared by Bahtiyar Kaba.

Table 4.2: Accuracy and RMS error for Poisson

	Accuracy 80%	Accuracy 90%	RMS
Depth 7	2.99 mm	4.02 mm	3.129
Depth 8	2.92 mm	3.91 mm	3.351
Depth 9	3.68 mm	4.98 mm	3.833

Source: This table has been prepared by Bahtiyar Kaba.

have red color (problematic) are more distinctive in (b). We will see if that makes any difference in numbers.

Back to the histogram in Figure 4.5, we see that a large portion of the error is distributed between -4 mm and 4 mm for depth 7. Similar histograms for other depth values can be found in Appendix A. This histogram indicates that we achieved a satisfactory results for our 3D scanning solution.

In Table 4.2 the accuracy values for 80 and 90 per cent thresholds and the RMS error value for each depth is shown. Again, we see that depth 7 has better quantitative results than the others. However, the small difference among them is not what we expected from our inspection of the meshes and error maps visually. The accuracy value for depth 9 is only marginally worse than the others (around 1mm for 90 per cent threshold). (Refer to Appendix A for more comparison results)

The reason why the accuracy and visual results differs might be due to the number of vertices in each mesh. As described in 3.2.1, the accuracy is calculated by setting a per cent threshold, finding the greatest distance in the per cent. Therefore, when the number of vertices large (as in model with depth 9) , the left-out points will also be numerous. Considering Figure 4.4 (c) , the red indicated regions could total up to 10 per cent of the entire mesh, therefore, will have no effect when the accuracy is calculated for 90 or 80 per cent. A similar explanation is valid for the RMS error. It is calculated by the mean of squared distances, and even if many noisy points will have larger values, the good points will drive the RMS lower. Considering these facts, it is obvious that some other error metric is required to objectively compare quantitative results of meshing algorithms.

5. CONCLUSION REMARKS

5.1 CONCLUSION

In this thesis study, we have worked building a 3D scanning solution alternative to expensive laser scanners. We proposed a setup with Kinect and a controlled rotating stage. We have satisfactory results which have accuracy values of around 5 millimeters for 90 per cent of our cloud mesh outputs.

Our contributions include a survey of active 3D range scanning where we have pointed the general architectural structure of 3D scanning. We have studied the algorithms for depth filtering, super resolution, alignment and surface reconstruction from point sets.

We have also carried out experiments relating to filtering and surface reconstruction and compared the output visually and quantitatively with accuracy, completeness and RMS metrics.

Based on our experimental results, we have proposed our scanning system. We have also evaluated how our scanning system output compares to the ground truth (laser scanner).

5.2 FURTHER RESEARCH

During this work, we have studied several problems in the field of 3D scanning and proposed our system after evaluation of some previous work. In this section, we discuss the points that we believe necessary for further study on this area.

3D reconstruction from color cameras and depth cameras have their advantages and disadvantages compared to each other. Color cameras are sensitive to illumination variations and does not give very reliable output, and depth cameras have low resolution output with noise. In the literature, color cameras are used to upsample the outputs of depth sensor and have positive results. However, a better approach can be to integrate their reconstruction results, that is, doing the optimization globally not on each frame. This could improve the overall output by allowing each to improve the other one's output iteratively.

Mean filtering and bilateral filtering methods have been applied on depth maps to improve the reconstruction quality. More clever filtering methods can be incorporated, however, as we see in our experiments, the effect of filtering the depth maps will be up to some point. The surface reconstruction step can be used to filter some noise with keeping details. Therefore, it is important to work on how these will affect the final output.

We used accuracy measure in our quantitative analysis. But, as we saw in our comparisons with different octree depths in Poisson algorithm evaluation, it might not be a good metric to interpret quality. We have also measured completeness and RMS and both suffered the same problem with accuracy. So, we claim it is important to come up with another metric that will quantify the output mesh more descriptively.

For alignment of the point clouds, we focused on rigid registration, i.e. no relative positions are not altered. However, even though filtering algorithms reduce noise to some extent, some of them still remains. After a rigid transformation of those noisy points, there occurs creases which negatively affect the reconstruction. Non-rigid deformations can help to solve this problem. A noisy region in an overlapping area can be fixed by the second cloud which has more correct data. Such algorithms can be integrated into the scanning procedure to see how they will work.

PUBLICATIONS

3D Reconstruction of Saltanat Gate in Dolmabahce Palace. Övgü Öztürk Ergün, Bo Zheng, Bahtiyar Kaba, Hüseyin İnan, Masataka Kagesawa, Katsuhi Ikeuchi. *International Conference on Cultural Heritage, Euromed 3-8 November 2014, Cyprus.*

Intelligent Interactive Applications for Museum Visits. Kemal Egemen Özden, Devrim Ünay, Hüseyin İnan, Bahtiyar Kaba, Özgü Öztürk Ergün. *International Conference on Cultural Heritage, Euromed 3-8 November 2014, Cyprus.*

A Framework for Low-cost 3D Reconstruction of Museum Artifacts, Bahtiyar Kaba, Ovgu Ozturk Ergun, IEEE Transactions on Multimedia(submitted).

A Survey on Active 3D Scanning Methods, Bahtiyar Kaba, Övgü Özturk Ergün, ACM Computing Surveys (to be submitted)

Technical Report for Turk Teleokom Collaborative Research Awards: Developing a Low-cost 3D Scanner for Digital Museums. Bahtiyar Kaba, Övgü Öztürk Ergün (2013)

REFERENCES

Books

Gonzalez, R., C., and Woods, R., E., 2007. *Digital Image Processing* 3. Edition Prentice Hall.

Grady, L., J., and Poimeni, J., 2010. *Discrete calculus*. Springer.

Hartley, R., and Zisserman, A., 2003. *Multiple view geometry in computer vision*. Cambridge university press.

Szeliski, R., 2010. *Computer vision: algorithms and applications*. Springer.

Watt, A., 1993. *3D computer graphics.* 2. Edition Boston, USA: Addison-Wesley Longman Publishing.

Journals

Besl, P., J., and McKay, N., D., 1992. A method for registration of 3-D shapes. *IEEE Transactions on pattern analysis and machine intelligence.* **14** (2), pp. 239-256.

Campbell, R., J., and Flynn, P., J., 2001. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding.***81** (2), pp. 166-210.

Curless, B., 1999. From range scans to 3D models. *ACM SIGGRAPH Computer Graphics*, **33**(4), pp. 38-41.

Farsiu, S., and Elad, M., and Milanfar, P., 2006.Multi-frame demosaicing and super-resolution of color image.*IEEE Transactions on image processing.* **15** (1), pp. 141-159.

Fischler, M., and Bolles, R., 1981. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography , Commun. ACM, 1981, 24, (6), pp. 381– 395.

Hoppe, H., and DeRose, T., and Duchamp, T., and McDonald, J., and Stuetzle, W., 1992. Surface reconstruction from unorganized points **26**(2), *ACM* pp. 71-78.

Khoshelham, K., 2011. Accuracy analysis of Kinect depth data.*ISPRS Workshop laser scanning.* **38** (5) , pp. 133-138.

Lorensen, W. E., and Cline, H. E., 1987. Marching cubes: A high resolution 3D surface construction algorithm. *ACM Siggraph Computer Graphics***21**(4) , pp. 163-169.

Lu, F., and Milios, E., 1997. Globally consistent range scan alignment for environment mapping. *Autonomous Robots* **4** (4), pp. 333-349.

Otsu, N., 1979. A threshold selection method from gray level histograms. *IEEE Transactions on systems, man., and cybernetics.***9** (1). pp. 62-66.

Xu, M., and Lu, J., 2012. Distributed RANSAC for the robust estimation of three-dimenional reconstruction. *Computer Vision, IET.* **6** (4), pp. 324-333.

Winslow, A. M., 1966. Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh. *Journal of computational physics*, **1**(2), pp.149-172.

Other Sources

Burrus, N., 2014 , Kinect Calibration, http://nicolas.burrus.name/index.php/Research/KinectCalibration [accessed 04 June 2014]

Chan, D., andBuisman, H., and Theobalt, C., and Thrun, S., 2008. A noise-aware filter for real-time depth upsampling.*In Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications* (M2SFA2 2008).

Chen, Y., Medioni, G., 1991. Object modeling by registration of multiple range images, *Proceedings of the 1991 IEEE International Conference on Robotics and Automation.* April 1991 California, USA, pp. 2724-2719.

Cui, Y., and Schuon, S., and Chan, D., and Thrun, S., and Theobalt, C., 2010. 3D shape scanning with a time-of-flight camera, *In proceedings of the IEEE conference on computer vision and pattern recognition.* June 2010 San Francisco, USA, pp. 1173-1180

Doumanoglou, A., andAsteriadis, S., andAlexiadis, D. S., and Zarpalas, D., and Daras, P., 2013. A dataset of Kinect-based 3D scans. *In IEEE 11th IVMSP Workshop*.June 2013, Seoul, Korea (pp. 1-4).

Fu, J., and Wang, S., and Lu, Y., and Li, S., and Zeng, W., 2012. Kinect-like depth denoising.*IEEE international symposium on circuits and systems(ISCAS)* May 2012 Seoul, pp. 512-515.

Huang, Q. X., and Adams, B., and Wicke, M., and Guibas, L. J., 2008. Non-Rigid Registration Under Isometric Deformations. *In Proceedings of Computer Graphics Forum***27** (5), pp. 1449-1457).

Kazhdan, M., Bolitho, M., and Hoppe, H., 2006. Poisson surface reconstruction.*Proceedings of the fourth Eurographics symposium on Geometry processing*. June 2006

Kinect for Windows Sensor Components and Specifications 2014.http://msdn.microsoft.com/en-us/library/jj131033.aspx [accessed 05 June 2014]

Konolige, K., and Mihelich, P., 2012, Technical description of Kinect calibration, http://wiki.ros.org/kinect_calibration/technical [accessed 04 June 2014]

Makadia, A., Patterson, A., and Daniilidis, K. 2006. Fully automatic registration of 3D point clouds. *In 2006 IEEE Conference on Computer Vision and Pattern Recognition.* **1** pp. 1297-1304.

Micos website , http://www.pimicos.com/web2/data/discontinued/1,5,150,rsp200.html [accessed 08 June 2014]

Newcombe, R. A., Davison, A. J., Izadi, S., Kohli, P.,Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., and Fitzgibbon, A., 2011. KinectFusion: Real-time dense surface mapping and tracking.*Inproceedings of the 10th IEEE international symposium on Mixed and augmented reality (ISMAR) 2011*, pp. 127-136.

Nguyen, V. C., and Izadi, S., and Lovell, D., 2012. Modeling Kinect sensor noise for improved 3d reconstruction and tracking. *Second international conference on 3D imaging, modeling, processing, visualization and transmission (3DIMPVT).* October 2012 Zurich, pp. 524-530.

Nielson, G. M., and Hamann, B., 1991. The asymptotic decider: resolving the ambiguity in marching cubes. *In Proceedings of the 2nd conference on Visualization'91*, pp. 83-91. IEEE Computer Society Press.

Park, J., and Kim, H., and Tai, Y. W., and Brown, M. S., and Kweon, I., 2011. High quality depth map upsampling for 3d-tof cameras. *In IEEE International Conference on Computer Vision (ICCV), 2011 IEEE* (pp. 1623-1630).

Rusinkiewicz, S., Levoy, M., 2001. Efficient variants of the ICP algorithm, *Proceedings of the third international conference on 3-D digital imaging and modeling.* 2001 Quebec City, Canada, pp. 145-152.

Schuon, S., and Theobalt, C., and Davis, J., and Thrun, S., 2009.LidarBoost: Depth superresolution for ToF 3D shape scanning. *In proceedings of the conference on computer vision and pattern recognition (CVPR).* June 2009 Miami, pp. 343-350.

Tomasi, C., and Manduchi, R., 1998. Bilateral filtering for gray and color images, *In the proceedings of the sixth international conference on computer vision.* January 1998 Bombay, India, pp. 839-846.

Vijayanagar, K. R., and Loghman, M., and Kim, J., 2012.Refinement of depth-maps generated by low-cost depth sensors, *2012 International Soc design conference.*November 2012, Jeju Island, pp. 355-358 .

Willow Garage, ROS-Kinect calibration: code-complete, 2010
http://www.ros.org/news/2010/12/kinect-calibration-code-complete.html
[accessed 01 June 2014]

Witkin, A. P., 1984. Scale-space filtering : a new approach to multi-scale description, *IEEE International conference on acoustics, speech, and signal processing (ICASSP'84).* March 1984, pp. 150-153.

Wu, C., and Wilburn, B., and Matsushita, Y., 2011. High-quality shape from multi-view stereo and shading under general illumination, *Proceedings of the 2011 IEEE conference on computer vision and pattern recognition.*June 2011 Providence, Rhode Island, pp. 969-976.

Zhao, M., and Tan, F., and Fu, C., and Tang, C., and Cai, J., and Cham, T. J., 2013. High quality kinect depth filtering for real-time 3d telepresence, *IEEE international conference on multimedia and expo (ICME).*July 2013, San Jose, USA, pp. 1-6.

Zhu, J., and Wang, L., and Yang, R., and Davis, J., 2008.Fusion of time-of-flight depth and stereo for high accuracy depth maps.*In IEEE Confrenece on Computer Vision and Pattern Recognition (CVPR)*. 2008, pp. 1-8.

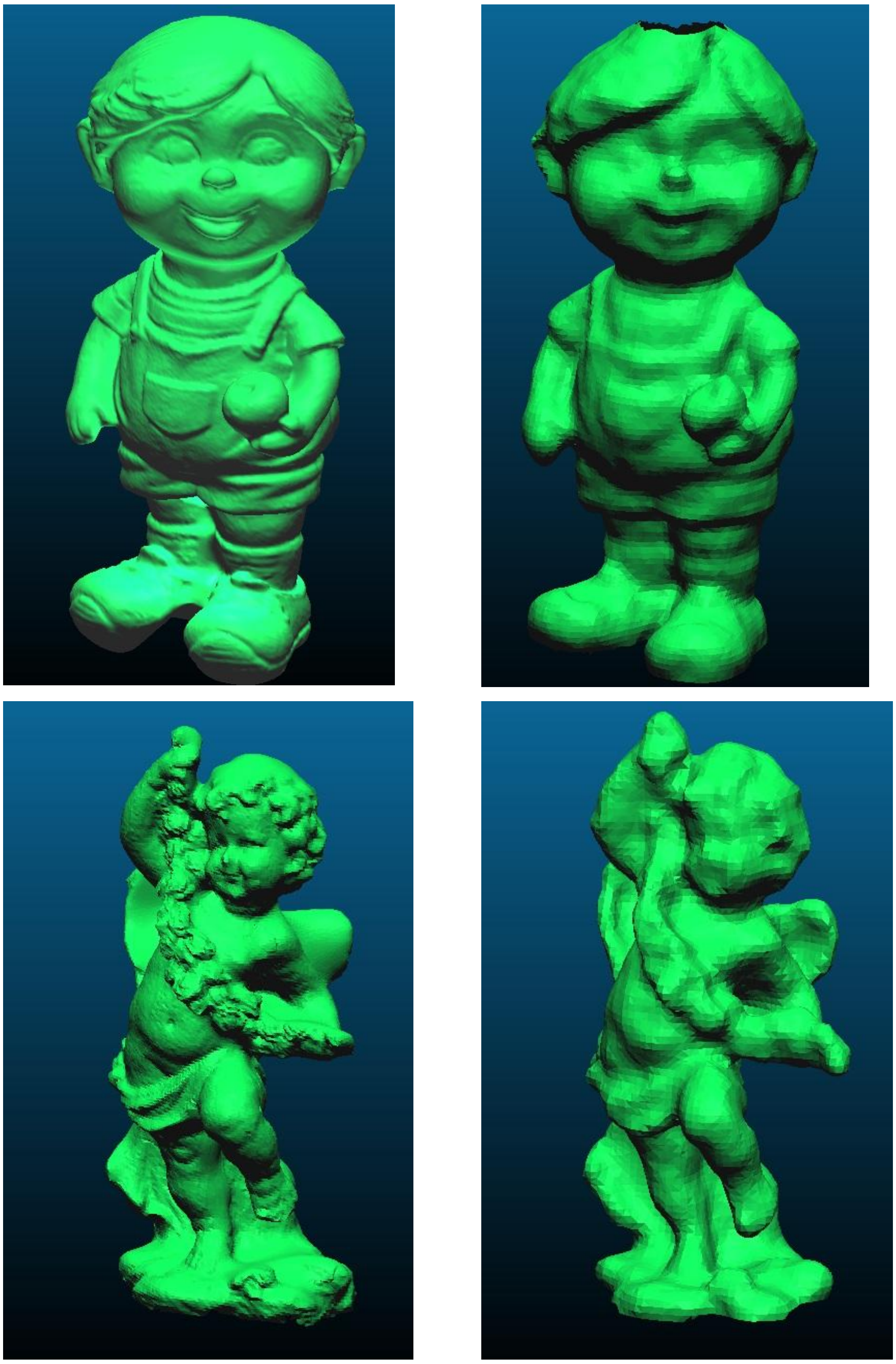
Cloud Compare Software website http://www.danielgm.net/cc/ [accessed 10 July 2014]

Meshlab Software website http://meshlab.sourceforge.net/[accessed 10 July 2014]

Point Cloud Library website http://pointclouds.org/ [accessed 10 July 2014]

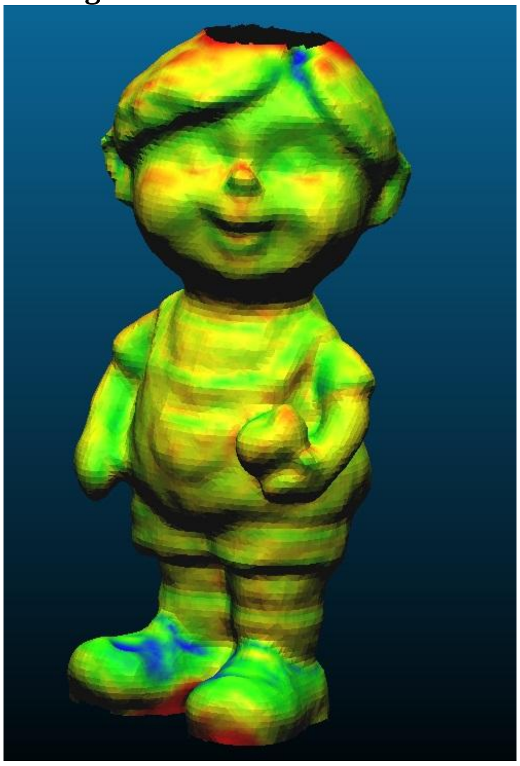
APPENDICES

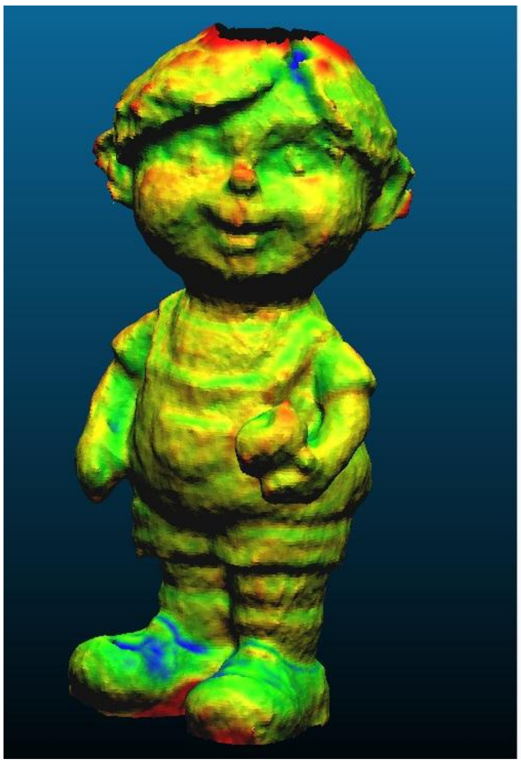
APPENDIX A-1. Laser scanning and our scanning results

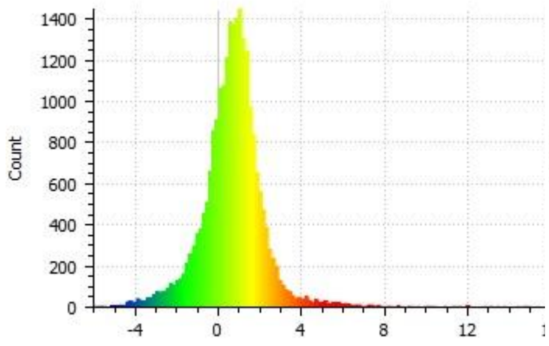


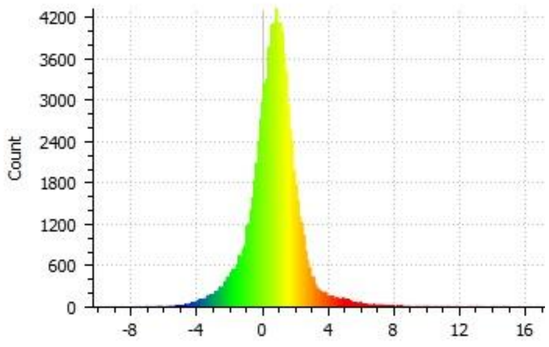
Laser scans (left) and our results for boy and angel objects(right)

APPENDIX A-2. Color coded error comparison for boy and angel object with histogram


Boy, octree depth 7

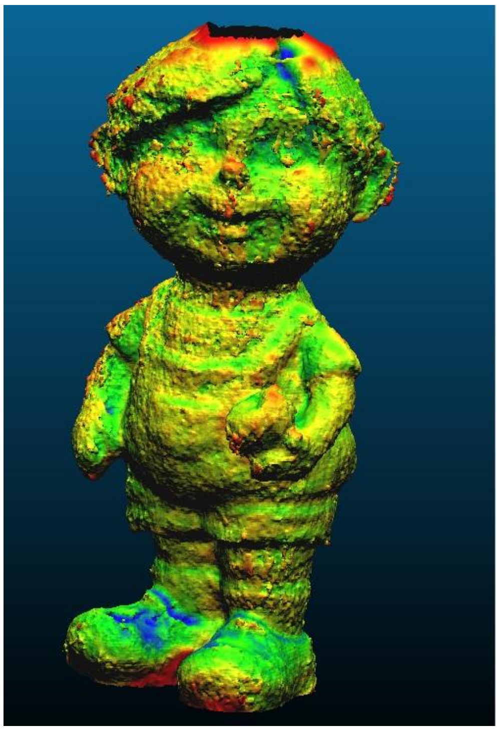

Boy, octree depth 8



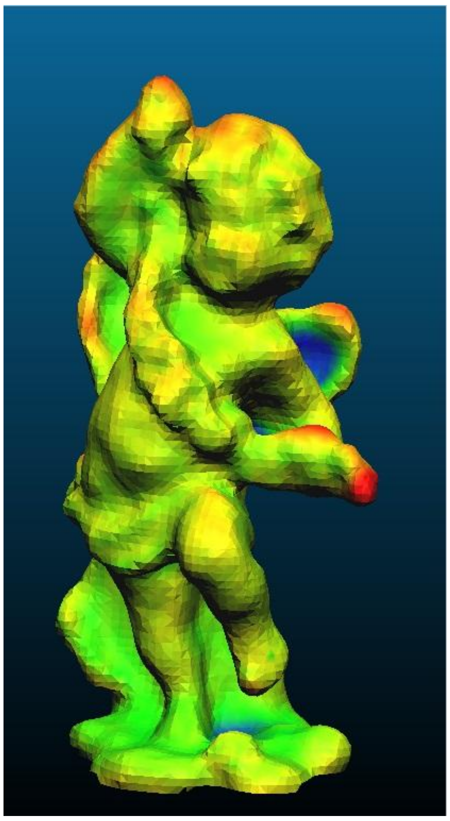


RMS	: 2.6872	RMS	: 2.7005
Accuracy 80%	: 2.03 mm	Accuracy 80%	: 1.97 mm
Accuracy 90%	: 2.88 mm	Accuracy 90%	: 2.62 mm

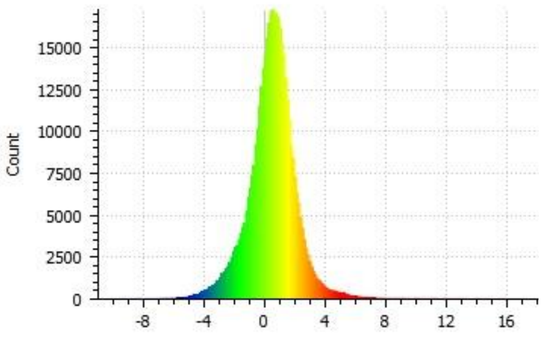
Color coded error comparison

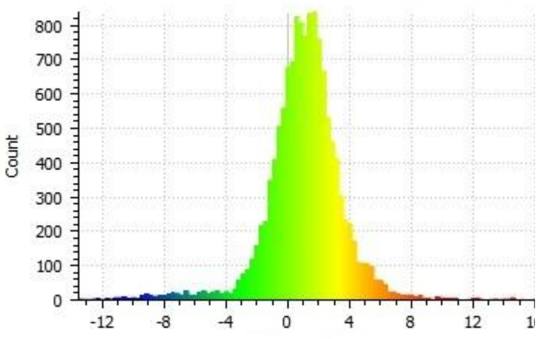


Boy, octree depth 9



Angel, octree depth 7

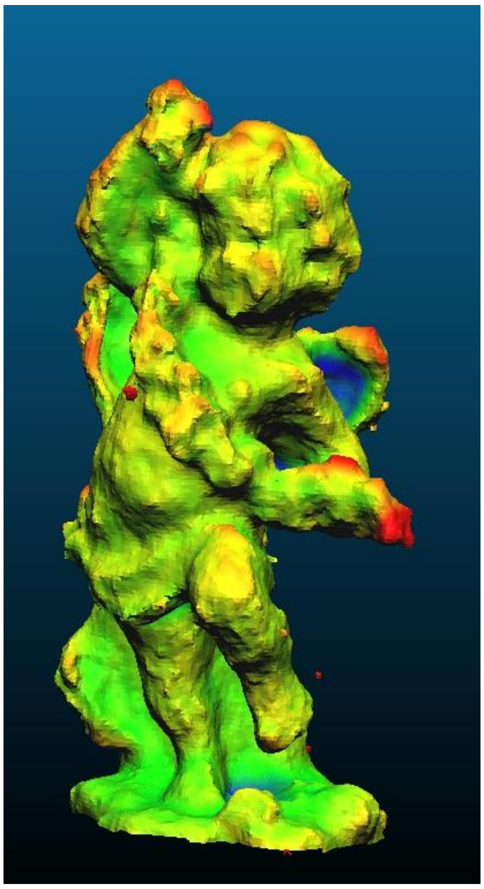


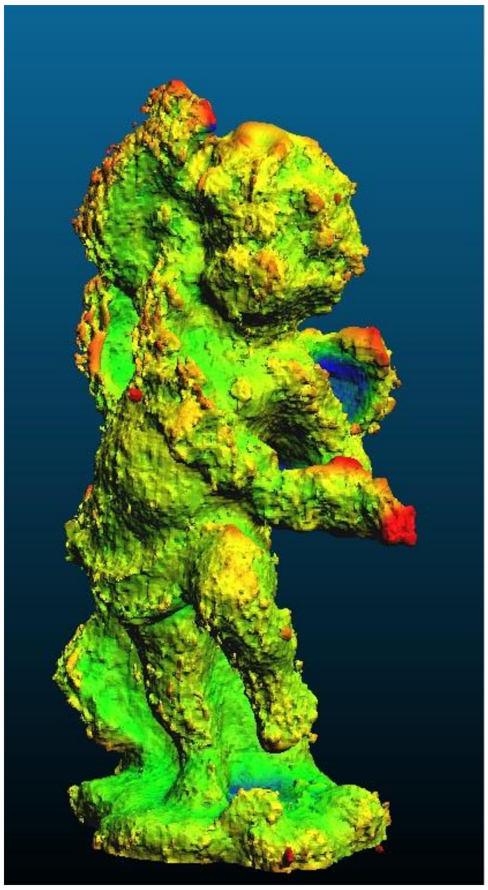


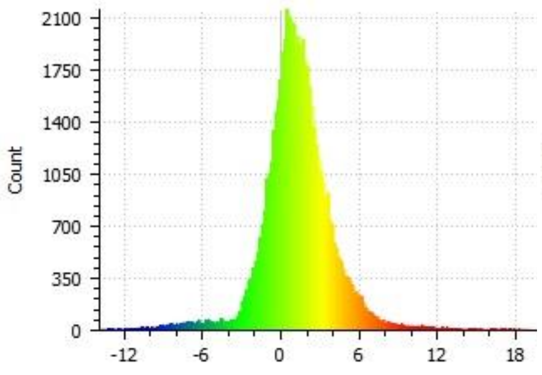
RMS	: 2.787
Accuracy 80%	: 2.18 mm
Accuracy 90%	: 2.92 mm

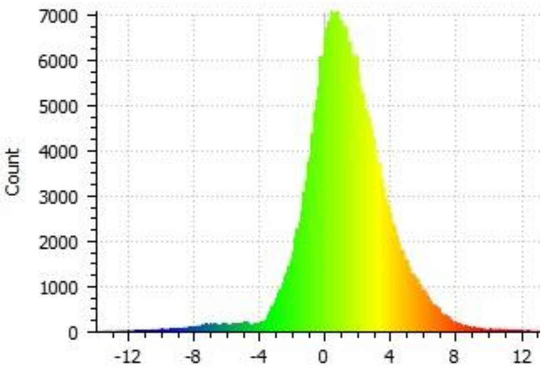
RMS	: 3.129
Accuracy 80%	: 2.99 mm
Accuracy 90%	: 4.02 mm

Color coded error comparison


Angel, octree depth 8


Angel, octree depth 9

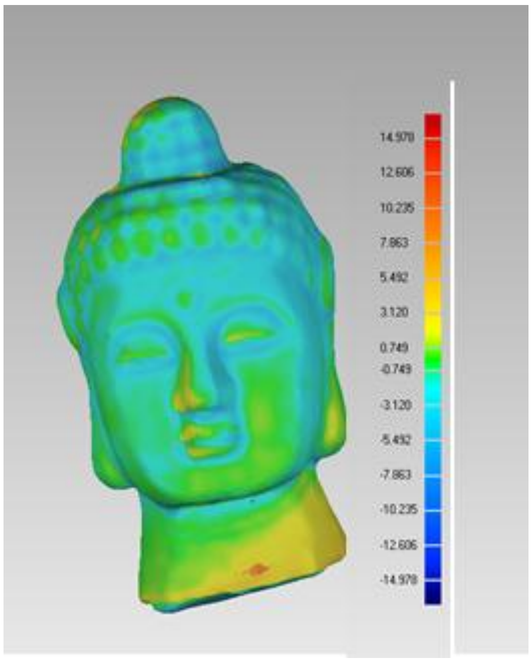


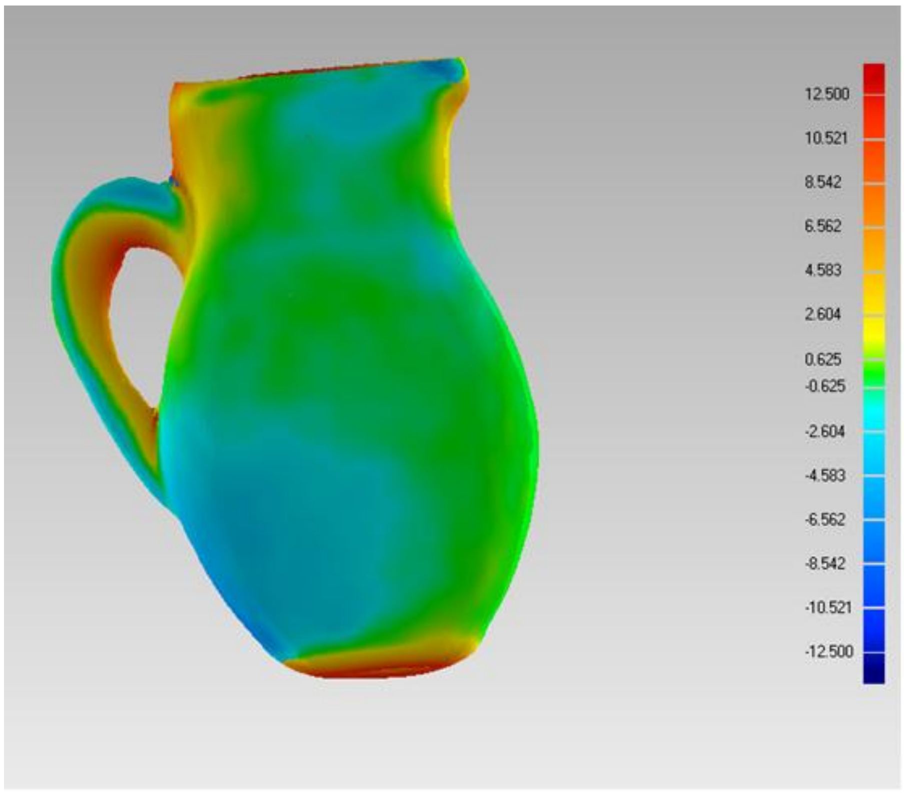


RMS	: 3.351	RMS	: 3.833
Accuracy 80%	: 2.92 mm	Accuracy 80%	: 3.68 mm
Accuracy 90%	: 3.91 mm	Accuracy 90%	: 4.98 mm

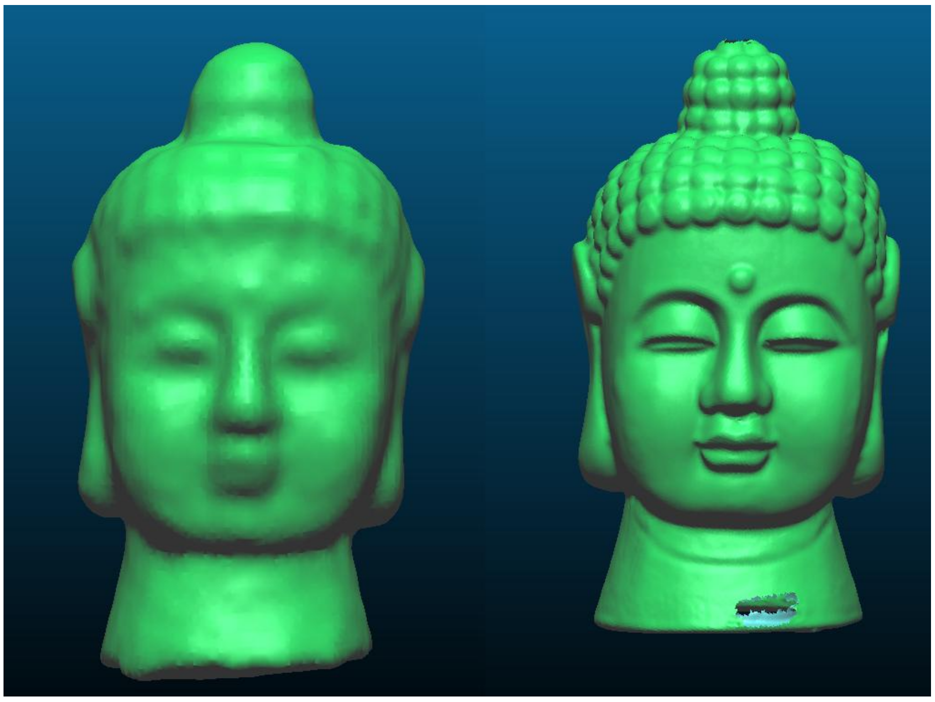
Color coded error comparison

APPENDIX A-3. Color coded error maps for Buddha and vase objects





APPENDIX A-4. Laser and our scanning result of Buddha



APPENDIX B-1. Pseudocode summary of our 3D reconstruction scheme

The indicated functions are assumed to be implemented such as NormalEstimation or ReadDepthMap. In our system, we implemented some of these functions and used the implementations in PCL library for some of them.

```
module Calibrate
BEGIN


      InitKinect
      depthMap <-- KinectAPI.AcquireDepthFrame
      removeBackground( depthMap )
      pointCloud <-- depthMap.Reproject( calibration_params)
      [plane1points, plane1params] <-- PlaneRANSAC(pointCloud)
      pointCloud.deletePoints(plane1points)
      [plane2points, plane2params] <-- PlaneRANSAC(pointCloud)
      axisParams <-- planeIntersect(plane1params, plane2params)

      return axisParams
END
```

```
module  ScanObject ( scanNumber, angle, axisParams )
BEGIN
      InitKinect
      FOR i <- 0 TO scanNumber
            depthMap <-- KinectAPI.AcquireDepthFrame
            removeBackground(depthMap)
            applyBilateralFilter(depthMap, 5, 3, 0.01)
            normalMap <-- estimateNormals(depthMap)
            pointCloudWithNormals <- reproject(depthMap,
                  normalMap,calibration_params)
            rotateCloud(pointCloudWithNormals, angle*i)
            pointCloudSet.push(pointCloudWithNormals)
            sendRotateCommandToCOMPort(angle)
      END
      return pointCloudSet
END
```

```
Module AlignSet(pointCloudSet)
BEGIN
    FOR i<- 0 TO pointCloudSet.size-1
        corr <- estimateCorrespondence( pointCloudSet[i],
            pointCloudSet[i + 1] )
        graph.insertEdge ( pointCloudSet[i], pointCloudSet[i + 1],
            corr )
    END

    corr<- estimateCorrespondence(pointCloudSet.last,
        pointCloudSet.first)
    graph.insertEdge(pointCloudSet.last, pointCloudSet.first, corr)

    ICPRegisterWithGraph(graph)

    return graph.ConcatenatedPointCloud
END
```

```
Module ReconsturctMesh(pointCloud, depth)
BEGIN
    downSamplePointCloud(pointCloud, 0.0001)
    octree<- generateOctree(pointCloud, depth)
    mesh <- Poisson(octree)

    return mesh
END
```

```
Main Module
BEGIN
    axisParams <- Calibrate
    pointCloudSet <- ScanObject(36, 10, axisParams)
    pointCloud <- AlignSet(pointCloudSet)
    mesh <- ReconstructMesh(pointCloud, 7)
END
```