ONE-TO-ONE SCALE MODELING FOR 3D PRINTING

A Thesis
by
MICHAEL HU YANG

Submitted to the Graduate School
Appalachian State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

August 2017
Department of Computer Science

ONE-TO-ONE SCALE MODELING FOR 3D PRINTING

A Thesis
by
MICHAEL HU YANG
August 2017

APPROVED BY:

Abstract

ONE-TO-ONE SCALE MODELING FOR 3D PRINTING

Michael Hu Yang
B.S., Appalachian State University
M.S., Appalachian State University

Chairperson: Rahman Tashakkori, Ph.D.

Current methods of 3D shape acquisition do not take into account the size of the object unless expensive laser scanning equipment is used. This thesis provides details on the design and implementation of creating a 3D model to size for 3D printing. The thesis explores a pipeline using depth information gathered from a Microsoft Kinect and various open source software and freeware. The algorithms and techniques that are utilized to create the 3D models are also described. VisualSFM's point cloud generator, Meshlab's Poisson surface reconstructor, and Microsoft's 3D Builder are some of the pipeline components used in the research. Results obtained at different stages of the process will be presented and discussed.

Acknowledgements

I would like to thank my mentor, Dr. Tashakkori for his endless support and guidance throughout my years at Appalachian State University. His wisdom and insight have helped shape my life here at Appalachian State University and has been invaluable to my success. I would also like to thank my committee members, Dr. Fenwick and Dr. McRae, for their time and feedback on the contents of this thesis. Thanks also to all my professors at Appalachian State University whose classes have equipped me with the skills I needed to complete this thesis and succeed in Computer Science. I would also like to thank the Department of Computer Science at Appalachian State University, the National Science Foundation, Lowe's Distinguished Professor Research Fund, and the Scholarships in Science, Technology, Engineering, and Mathematics (S-STEM) program for providing me the financial support and community of like-minded colleagues to grow with during my time at Appalachian State University. Finally, I would like to thank my family and friends I've made throughout the years at Appalachian State University for their encouragement and support.

Contents

List of Figures

List of Tables

Chapter 1 - Introduction

1.1 3D Scanning

Three-dimensional (3D) scanning is an active topic in various areas of computer science. The purpose of 3D scanning generally is to acquire information about an objects appearance to be used in constructing digital 3D models. The model should be as close as possible to the original object and preserve a degree of detail.

1.2 Motivation

Current methods of 3D shape acquisition do not measure the size of an object to scale. The most relevant methods are only able to measure distance/depth in addition to the shape [22]. While it is possible to determine the size of an object from this information, it is expensive in terms of manual efforts and equipment costs. With the arrival of the Microsoft Kinect, a cheaper alternative to expensive depth sensors was made available at the consumer-level. Creating models for 3D printing has long been a very tedious manual task. Models created using current 3D scanners were either not to the size or lacking sufficient details. Users are then required to edit models in modeling software manually to set the size and add detail. This thesis will try to address this tedious task.

1.3 Research Overview

This research builds upon the work done of Lembke and Gordet [19]. The goal of this research is to 3D print a one-to-one scale model of an object. This was accomplished through a pipeline using depth information gathered from the Kinect's depth sensor, Structure from motion (SFM) techniques to build a 3D model, and scaling the model using the depth information gathered previously. The term pipeline is used in its more general sense, a chain of processes where the output of one process is used as input for the next process.

1.4 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 examines recent works exploring the capabilities of the two versions of Kinect, 3D scanning, and the research that this thesis is built upon. Chapter 3 provides the theoretical foundations for the Kinect sensor, background subtraction, edge detection, size calculation, and 3D modeling. Chapters 4 and 5 discuss the implementation of these topics and the results of the previous methods. The performance at each stage of the process is discussed and sources of error analyzed and corrected. Chapter 6 summarizes the findings and concludes this thesis with avenues for future work introduced.

Chapter 2 - Literature Review

Three-dimensional (3D) data acquisition techniques have been highly researched in computer vision and 3D imaging. Numerous hardware sensors and software programs have been developed using different passive and active methods such as the Kinect v1 and Kinect v2. The Kinect v1 utilizes structured light and the Kinect v2 utilizes time-of-flight (ToF) methods [27]. This chapter discusses the previous work of researchers on the properties of Kinect v1, tools created to supplement the sensor, and properties of the newer Kinect v2 sensor. In addition, the work of Lembke and Gordet [19] is described.

In 2010, Microsoft introduced an affordable consumer-level depth sensing camera in the form of the Kinect v1 and its software development kit (SDK) which made it appealing for researchers. There have been several papers [3, 7, 13, 15] analyzing its properties as well as its application towards 3D object/scene modeling. Andersen et al. [7] describe an evaluation of the Kinect v1 depth sensor for computer vision applications. Three software frameworks, Microsoft SDK, OpenNI, and OpenKinect, are examined and OpenNI is chosen for being open-source, multi-platform, able to capture depth up to 9 meters, linearized, and having a smallest measurable depth difference of approximately 0.1m [7]. The Kinect v1 was found to have complications with reflective material, IR light interference, and notable structural noise. These complications with the Kinect v1 sensor can help decide if the Kinect depth sensor will be useful to an application [7, 13]. Soon after the Kinect SDK was released, Microsoft researchers released two papers introducing KinectFusion. KinectFusion is a system for accurate real-time mapping of complex and arbitrary indoor scenes using only a moving low-cost depth camera and commodity graphics hardware. All the depth data streamed from the Kinect v1 sensor is then fused into a single global implicit surface model of the observed scene in real-time using dense simultaneous localization and mapping (SLAM) while also tracking the sensor's agile motion using the depth data in each frame [14, 23].

Bueno et al. [9] conducted a metrological evaluation of KinectFusion and a standalone Kinect v1 sensor. A standard artifact consisting of 5 delrin spheres and seven aluminum cubes, all of varying sizes, was made. Accuracy and precision tests at different resolutions and ranges were then performed for a standalone Kinect v1 and a Kinect v1 with KinectFusion. In order to assess the quality of the models, geometric primitives (spheres and planes) were fitted to the point clouds using a least-squares algorithm. They established a proper comparison with the stand alone Kinect v1 sensor comparing the same features used by Gonzalez-Jorge et al. [12]. They found that the KinectFusion precision results were better than the Kinect v1 standalone. Precision values for KinectFusion were proportional to the resolution of the 3D model [9]. It was shown that using KinectFusion point cloud information to reconstruct bonsai plants can effectively model intricate detailed models [20].

In another study, a cart with two digital single-lens reflex (DSLR) cameras and two Kinect v1s working in tandem was pushed down a long hallway to map features. In the developed system, DSLR cameras were used to bridge the Kinect v1s and provide a more accurate ray intersection condition taking advantage of the higher resolution and image quality. Point cloud data was created using the Kinect v1 depth data and structure from motion (SFM) reconstruction was utilized to link and merge multiple point clouds to create an integrated point cloud. The process was successful in mapping indoor environments even in featureless areas [29].

Xu et al. [31] reported a system for object modeling using the Kinect v1 and a rotating platform. After the Kinect v1 is calibrated, the system is calibrated by placing the rotation platform into a unified coordinate system with the origin aligned at the center of the rotation platform. This allows the point clouds acquired from different views to be registered by the rotation transform easily. Poisson reconstruction is then used to recreate the surface mesh from the point cloud. This method is texture independent and does not need to rely on iterative closest point (ICP) [31].

Manikhi et al. [21] use a similar scanning set up with a rotating platform on which the object rests while scanning it with the Kinect. The platform is able to rotate incrementally using a stepper motor controlled by EiBot. A point cloud was generated for each object and stored in a database for further processing [21]. The solution proposed circumvents the point cloud registration task based on the fact that the transformation from one frame to the next is known with extremely high precision. The acquired 3D data is merged into a single noise-free model using a spherical transformation and analyzed.

Since the release of the Kinect v2 in 2014, there have been several papers analyzing its properties and calibration [11, 17, 18, 24, 28, 32] as well as several papers comparing it to its predecessor [12, 24, 27]. The Kinect v2 features an RGB camera with 1920x1080 resolution, a depth camera of 512x424 resolution, and a depth range of 0.5m to 4.5m utilizing a ToF principle [24]. ToF sensors measure the duration before a sent-out time modulated signal (usually light from a laser) returns to the sensor [22]. There are currently two categories of ToF sensors, pulsed and continuous wave. Pulsed ToF sensors measure the time delay between the sent and received signal. This requires a very accurate counting process that is not possible at room temperature. Continuous wave ToF sensors, such as the Kinect v2, take multiple samples, each phase-stepped by 90 degrees for a total of four samples per measurement. Then the phase offset is calculated and from that the distance is estimated [4, 11]. At the time, little analysis of the Kinect v2 was done, thus an accuracy evaluation of Kinect v2's depth sensor as well as calibration was required. Corti et al. [11] proposed a metrological characterization process for the Kinect v2 based on the guide to the expression of uncertainty in measurement. They evaluated the uncertainty for a 3D scene reconstruction based on different factors such as internal temperature of the Kinect, increase in depth and in radial coordinate, systematic uncertainty known as "wiggling error", different materials and surfaces, as well as reflectivity and concave 3D geometry [11]. Lachat et al. [18] confirmed the possible sources of error. They performed their own series of assessments for pre-heating time, material and colors, outdoor efficiency, as well as geometric and depth calibrations to address the "wiggling error" and radial distortions. The final conclusion is that the Kinect v2 is more accurate compared to the Kinect v1 [18]. A second assessment and calibration of the Kinect v2 was performed towards the specific use of close range 3D modeling. The depth sensor was analyzed using a checkerboard of gray scale squares of differing intensity values and the depth was examined for each square. The lower the intensities, the longer the distance measured. Deviations of up to 12mm were measured for the black squares. A sandstone balustrade fragment was scanned using the Kinect v2 and KinectFusion. The accuracy was in an order of magnitude of 1 cm which was considered satisfactory although many improvements could still be considered to improve the accuracy [17].

Lembke et al. [19] proposed a pipeline for 3D reconstruction via photogrammetry, specifically SFM. SFM is the strategy of connecting multiple image frames in sequence using passive stereo [22]. Passive stereo methods use two cameras to view a scene, or if the scene is static then a single camera at two positions could take the images in sequence. The SfM concept is analogous to biological vision in which humans and other living creatures can acquire a lot of information regarding the 3D structure of a scene as information is sensed over images within a period of time [26]. When multiple views are available, the camera does not need to be calibrated beforehand with dedicated calibration patterns [22]. This makes SFM an ideal 3D acquisition strategy. Using a combination of an IPhone 6 camera and Nikon Coolpix camera, image sequences were obtained of various objects and their point clouds examined for 3D printing.

Previous works have all examined the intrinsic uncertainties and sources of errors of the Kinect v2. There are some previous works that use the same set up of a rotational platform to scan an object [21, 31], but none of them for the purpose of 3D printing. This thesis will build on previous work by [19] to reconstruct 3D models of objects, specifically for 3D printing purposes.

Chapter 3 - Theoretical Background

3.1 Microsoft Kinect 2

In November 2013, Microsoft released the Kinect v2 sensor, henceforth referred to as simply Kinect, for Windows and Xbox One, improving upon the Kinect v1 sensor for Xbox 360 to provide high resolution images and depth information. With these features, Kinect can function as a low cost alternative to expensive 3D laser scanners. Knowing the characteristics of the sensor is an important first step in 3D scanning, as all subsequent steps such as background subtraction, edge detection, and size calculation are dependent on accurate readings from the sensor.

3.1.1 Specifications and Characteristics

The Kinect uses infrared (IR) projectors and an IR camera to capture depth images. The main characteristics of the Kinect are outlined in Table 3.1. Kinect offers RGB and depth capture which will be used to shape the 3D model and calculate the actual size of the object, respectively. Table 3.2 outlines IO parameters of the RGB and IR cameras estimated during camera calibration in [24].

The presence of radial distortion manifests in the "fish-eye" effect the further a pixel is from the principal point as shown in Figure 3.1. The magnitude of distortion increases as the distance from the principle point increases. The radial distortion

Table 3.1: Kinect 2 characteristics

RGB image resolution (pixel)	1920 x 1080
RGB pixel size (µm)	3.1
Depth image resolution (pixel)	512 x 424
Depth pixel size (µm)	10
Depth sensing principle	Time of flight
Max depth distance (m)	8
Min depth distance (m)	0.5
Horizontal FoV (degrees)	70
Vertical FoV (degrees)	60
Price ($)	199

Table 3.2: IO parameters of RGB and IR cameras estimated during camera calibration

	RGB Camera		**IR Camera**	
	Value	**Std. Dev**	**Value**	**Std. Dev**
Focal length (mm)	3.291	1.0e-3	3.657	5.2e-4
Principal point x (mm)	-0.005	5.6e-4	0.032	3.5e-4
Principal point y (mm)	-0.016	6.9e-04	0.033	3.9e-4
Radial Distortion Parameters				
$\mathbf{K_1}$ (mm^{-2})	3.823e-3	3.8e-5	-6.510e-3	2.7e-5
$\mathbf{K_2}$ (mm^{-4})	3.149-4	3.8e-6	1.205e-3	3.8e-6
Tangential Distortion Parameters				
$\mathbf{P_1}$ (mm^{-2})	2.332e-4	2.0e-5	1.377e-4	8.0e-6
$\mathbf{P_2}$ (mm^{-2})	-5.152e-4	2.1e-6	1.589e-4	9.2e-6

parameters in Table 3.2 are used to find the corrected coordinates for a given pixel, Equation (3.1), where **r** is the radial distance from the principal point and **K** is the radial distortion parameter. The tangential distortion occurs if the camera lens are not perfectly parallel to the imaging plane caused by misalignment and the magnitudes of distortion can be seen in Figure 3.2. This distortion can be corrected using Equation (3.2), where **P** is the tangential distortion parameter.

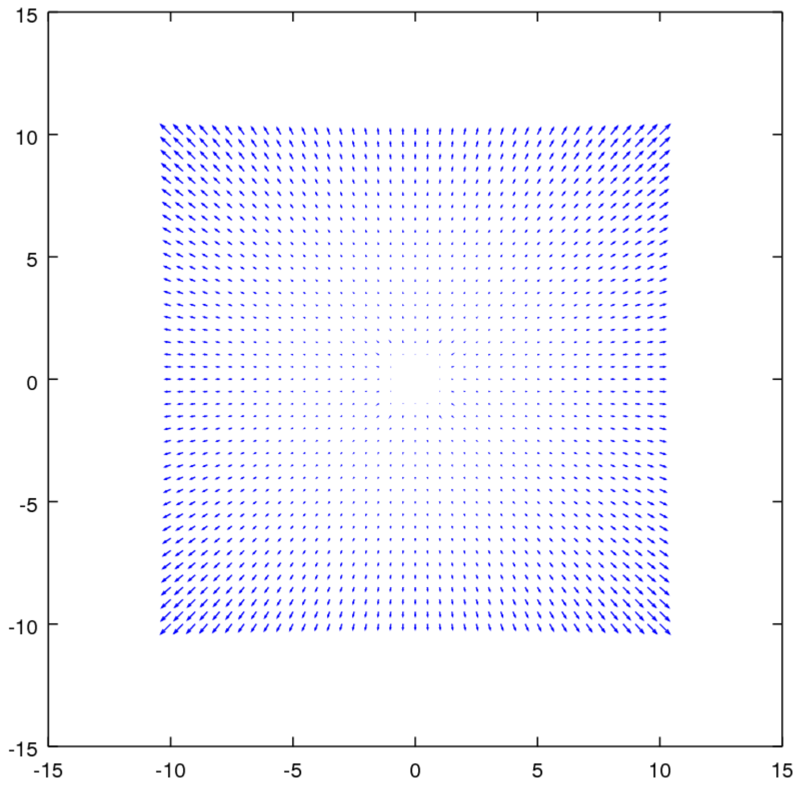


Figure 3.1: Radial distortion magnitudes

$$\mathbf{r} = \sqrt{\mathbf{x}^2 + \mathbf{y}^2}$$
$$\mathbf{x_{corr}} = \mathbf{x}(1 + \mathbf{K_1 r^2} + \mathbf{K_2 r^4}) \tag{3.1}$$
$$\mathbf{y_{corr}} = \mathbf{y}(1 + \mathbf{K_1 r^2} + \mathbf{K_2 r^4})$$

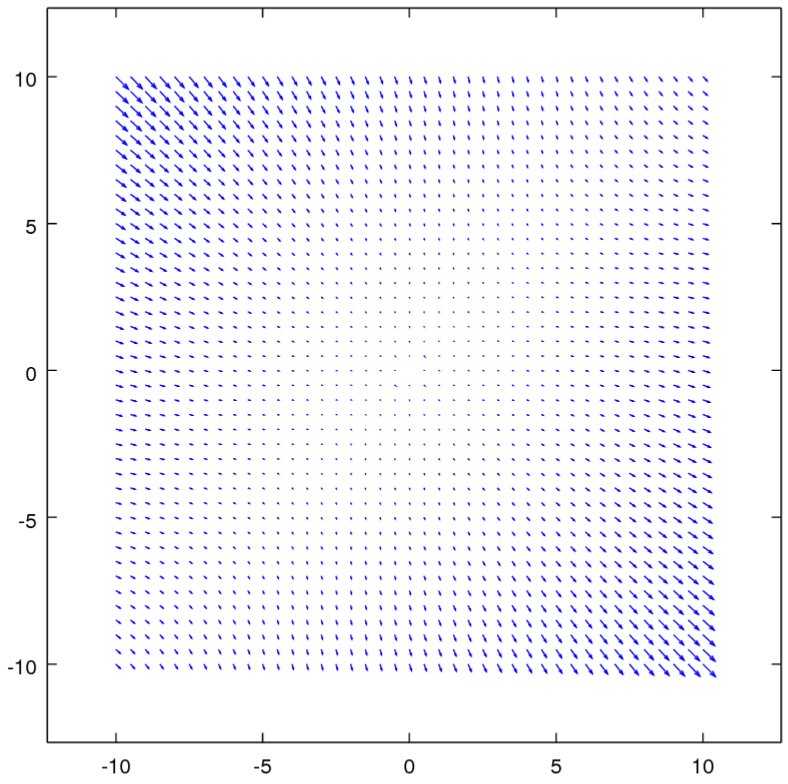


Figure 3.2: Tangential distortion magnitudes

$$\begin{aligned} \mathbf{x_{corr}} &= \mathbf{x} + (2\mathbf{P_1xy} + \mathbf{P_2}(\mathbf{r^2} + 2\mathbf{x^2})) \\ \mathbf{y_{corr}} &= \mathbf{y} + (\mathbf{P_1}(\mathbf{r^2} + 2\mathbf{y^2}) + 2\mathbf{P_2xy}) \end{aligned} \tag{3.2}$$

3.1.2 Time of Flight

The Kinect uses the Time of Flight (ToF) method to acquire a 3D map of depths through the diffusion of light on the scene. The two measuring principles of ToF sensors are pulsed and continuous wave and the Kinect uses the latter. In a pulsed ToF sensor, the time delay between the sent and received pulses are measured with a fast counter synchronized with the emitted signal that is converted into a distance. This requires a very accurate counting process and is not able to be achieved at room temperature. In the continuous wave method, an emitted light is modulated with a square wave with frequency (f) in the range of 10-100MHz. Figure 3.3 shows four samples, each phase-stepped by 90° of the reflected light that are captured by the IR sensor to obtain the phase offset (ϕ). The distance (d) is computed per pixel using Equation (3.3), where the measurements $Q1$, $Q2$, $Q3$, and $Q4$ are from the shaded areas from the phase-stepped samples C1, C2, C3, C4 in Figure 3.3, respectively.

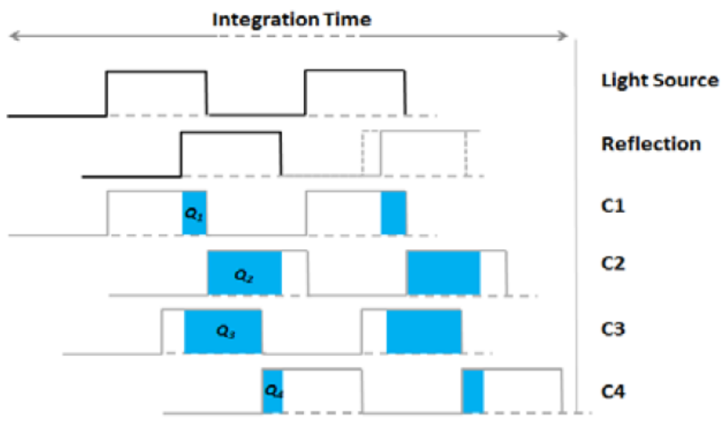


Figure 3.3: Four phase-stepped samples [4].

$$\begin{aligned}\phi &= \arctan\left(\frac{Q3 - Q4}{Q1 - Q2}\right) \\ d &= \frac{c}{4\pi f}\phi\end{aligned} \tag{3.3}$$

3.2 Background Subtraction

In image processing, a common initial step in classification or recognition problems is background subtraction, also known as foreground detection. Generally, the area of interest of an image consists of objects that are in the foreground. In the case of a single static camera aimed at a static reference frame, the foreground is the object that is introduced into the camera's field of view and was not present in the reference frame. The gain of background subtraction is a reduction in noise due to clutter and other objects of insignificance that may be present in the frame.

3.2.1 Pixel Subtraction

In a static camera and static reference background, noise caused by variable lighting and moving objects are not a major concern. We can use a straight forward process of pixel subtraction where each pixel of the reference frame is subtracted from the corresponding pixel of the frame where the object is present.

3.3 Edge Detection

Edge detection is useful for image segmentation and data extraction in applications such as image processing, computer vision, and machine vision. It is an image processing technique that finds boundaries of objects/features in an image by examining discontinuities between the brightness of pixels. These discontinuities in brightness are likely to correspond with discontinuities in depth, surface orientation, variations in illumination, and changes in material composition such as color [8]. In the ideal case, lines and connected curves may constitute the boundaries of the object and reduce the area of significance to be processed. Hurdles that must be overcome include fragmentation meaning edge curves that are not connected to other curves, gaps in edges, and false edges meaning detected edges that are not significant adding noise to the data [5].

3.3.1 Canny Edge Detection

In 1986, John F. Canny developed an edge detection algorithm after he found that many diverse applications utilizing edge detection were relatively similar in their requirements. These applications needed to satisfy three criteria:

1. Low error rate

2. Edge points should accurately localize on the center of the edge

3. Each edge should be marked only once

When these criteria are satisfied, the process of Canny Edge Detection can be broken down into 5 steps [1]:

1. Filter out noise using a Gaussian filter. A typical filter of kernel size 5 is given in Equation (3.4):

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{3.4}$$

2. Find the intensity gradient of the image. An edge may point in a horizontal, vertical, or diagonal direction so a Sobel operator is used to return the first derivative in the horizontal and vertical direction by applying a convolution mask in the horizontal and vertical direction as described in Equation (3.5) on G_x and G_y, respectively. From this, the edge gradient (G) and direction (θ) can be found

using Equation (3.6).

$$
\begin{aligned}
G_x &= \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \\
G_y &= \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}
\end{aligned}
\tag{3.5}
$$

$$
\begin{aligned}
G &= \sqrt{G_x^2 + G_y^2} \\
\theta &= \arctan\left(\frac{G_y}{G_x}\right)
\end{aligned}
\tag{3.6}
$$

3. Non-maximum suppression is applied. This is used to thin edges to satisfy the third criterion since after the gradient calculation, the edges may still be quite blurred. This algorithm compares each point suppressing each gradient value to 0 except the local maximum which indicates the sharpest change in intensity.

4. Double threshold calculation. After non-maximum suppression is applied, there will still be edges caused by noise. An upper and lower threshold is picked and any values greater than the upper threshold will be marked as strong, any values below the lower threshold will be suppressed, and values in between will be marked as weak.

5. Edge tracking by hysteresis. For each weak pixel, the surrounding 8 pixels are analyzed and if a strong pixel is present in the surrounding area, the weak pixel is preserved and included in the final image.

3.4 Size Calculation

Many areas of image processing aim to measure the size of objects. This can be done in many ways such as through the use of a reference object of known size for comparison or calculating the size if the focal length of the lens and distance from the object is known. Using a reference object requires that the objects are co-planar and the lens is parallel to this plane. If the focal length of the lens is known and the distance to two points on the object are also known, the distance between those two points can be found through trigonometry. These two points can be picked in such a way that the distance between them is also a significant dimension of the object such as length, width, or height.

3.4.1 Pinhole Camera Model

The pinhole camera model describes the mathematical relationship between a 3D point and its 2D corresponding point on an image plane in Figure 3.4, where P is the 3D point, Q is the corresponding 2D point on the image plane, f is the focal length of the camera lens, and d is the distance of the object from the lens. Because this forms a pair of similar triangles, the relationship in Equation (3.7) holds.

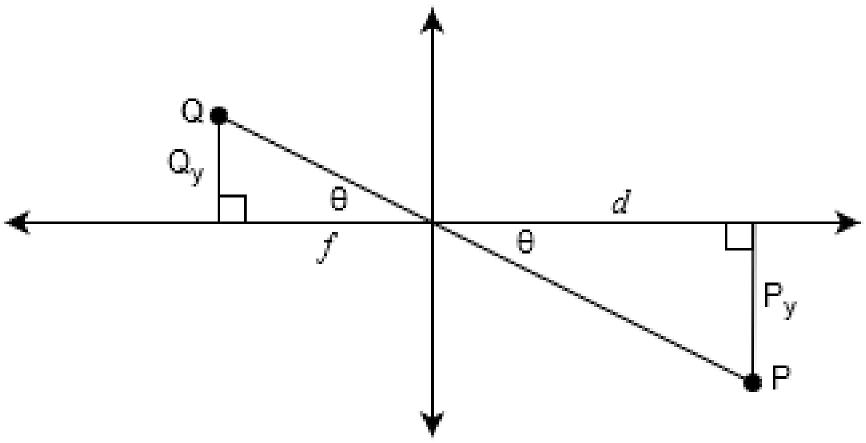


Figure 3.4: Pinhole camera model

$$\frac{Q_y}{f} = \frac{P_y}{d} \tag{3.7}$$

By knowing the focal length, the distance, and the number of pixels the object covers on the image plane (Q_y), we can solve for P_y as Equation (3.8).

$$P_y = \frac{Q_y \cdot d}{f} \tag{3.8}$$

Figure 3.5 illustrates two points of an object at different distances from the lens. In this figure, P_y is the distance between the two points if they were both at the same distance (d_1) from the lens, d_2 is the actual distance of A from the sensor, and l is the real world length between P and A. We can solve for (l) by using the Pythagorean theorem in Equation (3.9) and plugging in Equation (3.8).

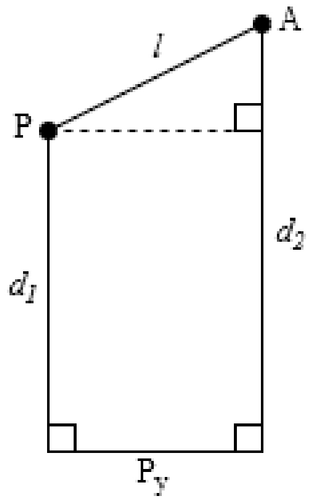


Figure 3.5: Length l between points P and A at different distances from the lens

$$l = \sqrt{P_y^2 + (d_2 - d_1)^2}$$
$$l = \sqrt{\left(\frac{Q_y \cdot d_1}{f}\right)^2 + (d_2 - d_1)^2} \tag{3.9}$$

3.5 3D Modeling

The digitization of real world objects is growing rapidly with applications in design, entertainment, and many other fields [25]. Sections 3.5.1 and 3.5.2 discuss methods of generating a 3D point cloud and a surface reconstructed mesh from the point cloud, respectively.

3.5.1 Structure from Motion / VisualSFM

Structure from motion (SFM) is a photogrammetric range imaging technique used to determine 3D shape from a series of 2D images. If a scene is static, two images may be taken in sequence at different positions to build correspondence. By tracking feature points over time between the images, a 3D model can be produced. Several steps can be taken to improve the result. Images can be taken in shorter intervals so that finding correspondence is easier. Having more camera views will lead to building a more complete model by increasing the number of possible feature points. If multiple view points are available, the camera no longer needs to be calibrated before-hand as internal and external parameters can be extracted from the images themselves using a self-calibration technique [22].

Finding correspondence between images typically uses the scale invariant feature transform (SIFT) algorithm. The SIFT descriptor is invariant to translations, rotations and scaling transformations in the image domain, perspective transformations, and illumination variations. The algorithm can be separated into four major stages of computation as follows [6]:

1. Scale-space extrema detection: The first stage of computation searches over all scales and image locations. It is implemented efficiently using a difference-of-

Gaussian function to identify potential interest points that are invariant to scale and orientation.

2. Keypoint localization: At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.

3. Orientation assignment: One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

4. Keypoint descriptor: The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

VisualSFM is a GUI based program utilizing SIFT for feature detection and matching, creating sparse and dense point clouds of these keypoints [30, 33, 34]. VisualSFM takes as input a sequence of images and computes correspondence between these points. The points are plotted in a 3D space to form a sparse point cloud. Clustering views for multi-view stereo (CMVS) is used to form a denser point cloud. Many multi-view stereo (MVS) algorithms do not scale well to a large number of input images (lack of computational and memory resources). CMVS takes as input the results of SFM processing, then decomposes the input images into a set of image clusters of manageable size. An MVS software can be used to process each cluster independently and in parallel, where the union of reconstructions from all the clusters should not miss any details that can be otherwise obtained from the whole image set [2]. From this point cloud, a surface mesh can be reconstructed using the process described next in Section 3.5.2.

3.5.2 Poisson Surface Reconstruction / Meshlab

The topic of surface reconstruction is well covered in computer graphics [10, 16]. It allows fitting of scanned data, filling of surface holes, and remeshing of existing models, though there are many difficulties in practice. The point sampling is often non-uniform, accessibility constraints during scanning may leave regions blank, and positions are noisy due to scanning misregistrations and sampling inaccuracies. Poisson surface reconstruction aims to fix these issues by expressing surface reconstruction as a solution to the Poisson equation shown in Equation (3.10), where $\triangledown^2$ is the Laplace operator, X is the scalar function whose gradient best approximates a vector field $\overrightarrow{V}$ defined by the samples [16].

$$\triangledown^2 \cdot X = \triangledown \cdot \overrightarrow{V} \tag{3.10}$$

Meshlab [10] is an open source mesh processing tool. It was created with the following objectives in mind:

- Ease of use: The tool should be designed so that users without high 3D modeling skills can use it (at least for the most basic functionalities).
- Depth of use: Tools should be designed so that an advanced user can extend functionalities.
- Single mesh processing oriented: The system should try to stay focused on mesh processing instead of mesh editing and mesh design.
- Efficiency: 3D scanning meshes can easily be composed by millions of points, so the tool should be able to manage them.
- Sustainability: The project should be able to grow and be self sustainable for a number of years.

Meshlab is able to take a point cloud output from an MVS program. The user can manually remove any noisy points from the cloud and generate a mesh using Poisson surface reconstruction.

Chapter 4 - Methodology

4.1 Introduction

Scanning an object and creating a 3D model of the correct size happens in a pipeline over several stages. There are 4 stages that were taken in this thesis to produce a finished model: data acquisition, background subtraction, point cloud generation, and surface reconstruction. An overview of this process is given in Figure 4.1.

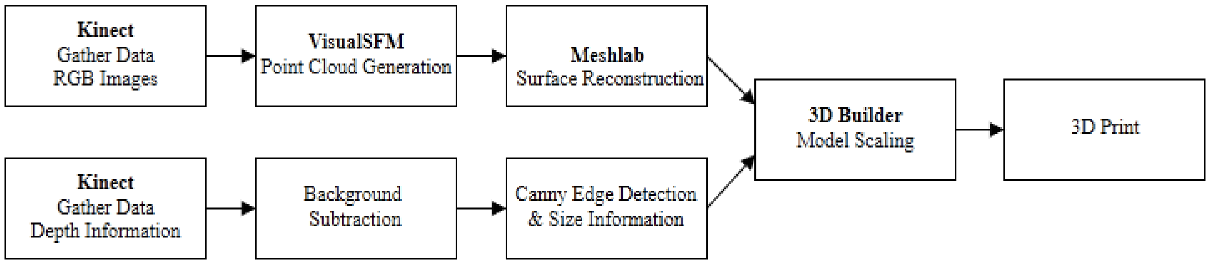


Figure 4.1: The 3D reconstruction of objects process pipeline

4.2 Data Acquisition

A Kinect was used to acquire RGB and depth images of 1920x1080 and 512x424 resolution, respectively. Depth information was also recorded per pixel of the depth image in a matrix. This matrix holds the distance in millimeters of the real-world point from the Kinect depth sensor at the corresponding pixel coordinates.

A series of RGB images were taken while circling the object for the purpose of point cloud generation which is discussed in Section 4.4. Distinct objects in the background will allow VisualSFM to match sequential images. Taking images of the object as it rotates on a platform from a static view also suffices, provided that the background has no discernible feature and the object has many. This method will provide a cleaner point cloud without background noise, whereas the former method will require cleaning up the point cloud of the 3D background space. In this thesis, an object was placed on a wooden stool in the middle of a room and a Kinect was used to capture images as it was moved around the object. These images were captured at different angles and distances from the object.

Multiple matrices containing the depth information were also taken from a static view containing just the background and again with the object present, as shown in Figure 4.2. These are used in the background subtraction method to isolate the object's depth information in Section 4.3. The Kinect was placed 0.5m away from the stand. For each view, 50 frames were captured as it was found that any more would not significantly improve quality. Frames were captured at 3 frames per second (fps) to allow time to write the depth matrices to a file.

Canny edge detection was used on the background subtracted image to highlight an outline of the object. The user can adjust the minimum threshold for detecting edges. A higher value means only distinct edges are shown. This can be used to get rid of false edges due to excess noise. Depth values of pixels on the left and right sides of the object were selected from the edge detection image and the distance between the two

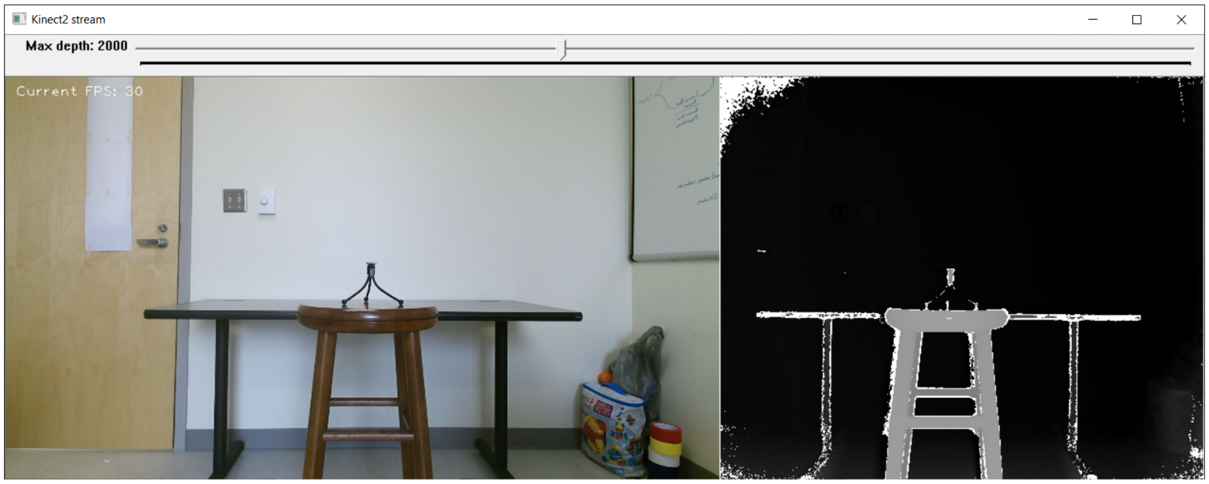


Figure 4.2: RGB and depth background for use in the background subtraction process

points was determined using the corresponding depth matrices and Equation (3.9). This corresponds to a real-world dimension of the object that will be used to scale the model to size in Section 4.5.

4.3 Background Subtraction

The background subtraction method used in this thesis is a simple pixel subtraction described in Section 3.2. This background subtraction stage is important for cleaning up the image of any unwanted objects and allows easier information to be gathered from the cleaned-up image.

First, the depth images and information captured by the Kinect feature large amounts of static and fluctuation at points where the Kinect was unable to consistently read the depth. To clean this up, 50 frames of the background image and object were captured and stored in a database. An average depth was calculated per pixel, where null values were omitted from the calculation. A matrix with the averaged values of each pixel was saved and a gray-scale image of the average values was created. This was done for both the background frames and frames with the object present. The averaged background image was then subtracted from the averaged object image giving an image of just the object present and a filter was applied to smooth the image. The main goal of some filters is to smooth an input image by removing noise. However, sometimes the filters not only dissolve the noise, but also smooth away the edges. To avoid this, we used a bilateral filter. The averaged image was used in the edge detection stage and the averaged matrix provided the values returned by the edge pixel selector and used in the size calculation.

4.4 Point Cloud Generation

Point cloud generation was done to create sparse and dense 3D point clouds using the SFM technique discussed in Section 3.5.1. For this stage, the series of RGB images taken by the Kinect in the data acquisition stage (Section 4.2) were loaded into the VisualSFM program. A SIFT algorithm was used on each image to detect discernible features that were used to match with other images. If images were taken while moving around the object, distinct features in the background would help match up sequential images if the object has none itself. The SIFT feature descriptor is invariant to uniform scaling, orientation, illumination changes, and partially invariant to affine distortion which allows the matching of images taken at different angles, distances, and tilts which covers most errors that may arrise due to an unsteady hand. With the focal length of the lens in mind, each pair of images were placed in the right orientation and distinct feature points were triangulated and plotted in a 3D space. Doing this for all matches, a sparse point cloud was generated. A dense point cloud was generated from these features using CMVS and saved for use in surface reconstruction. If the sparse point cloud contained many features, the dense point cloud would be saved in multiple layers of a project bundle.

4.5 Surface Reconstruction

Meshlab is used to clean up the dense point cloud and create a surface reconstruction of the object. Opening the project will load the sparse point cloud initially. This layer can be deleted since this thesis requires the richer dense point clouds. If more than 1 layer is loaded in, these dense point cloud layers can be merged into a single layer to be processed as a whole. Noise and unimportant points can be selected and removed from the layer. Once the dense point cloud is sufficiently cleaned, a Poisson surface reconstruction algorithm can be applied to the point cloud. The Poisson algorithm takes in 4 parameters that will affect the quality and characteristics of the model:

- Octree Depth: The depth of the Octree used for extracting the final surface is set with a suggested range between 5-10. Higher numbers mean higher precision in the reconstruction but also higher processing times.
- Solver Divide: This argument specifies the depth at which a block Gauss-Seidel solver is used to solve the Laplacian equation. This parameter helps reduce the memory overhead at the cost of increased reconstruction time.
- Samples per Node: This point value specifies the minimum number of sample points that should fall within an octree node as the octree construction is adapted to sampling density. For noise-free samples, small values in the range 1.0-5.0 are used. For more noisy samples, larger values in the range 15.0-20.0 may be needed to provide a smoother, noise-reduced, reconstruction.
- Surface Offsetting: This point value specifies a correction value for the isosurface threshold that is chosen.

Once this process is finished and a surface mesh is created, it can be saved and loaded into Microsoft Window's 3D Builder app. Meshlab does not save the information about the size of the mesh and when importing into 3D Builder, a prompt of different measurement units will appear. Selecting "inches" gives the closest initial size of the object, but the dimension must be adjusted to the size measured in Section 4.2. 3D

Builder provides a measurement tool that can be used for this stage. The model is then saved as an .stl file which retains the size information and can be printed.

Chapter 5 - Results

5.1 Background Subtraction

Background subtraction is the first stage in attaining the correct dimension of an object. Figure 5.1 shows the image that will be considered as the background along with a depth image for one frame that will be used in the average. Figure 5.2 shows the image with the object added into the frame and the averaged background image will be subtracted from it.

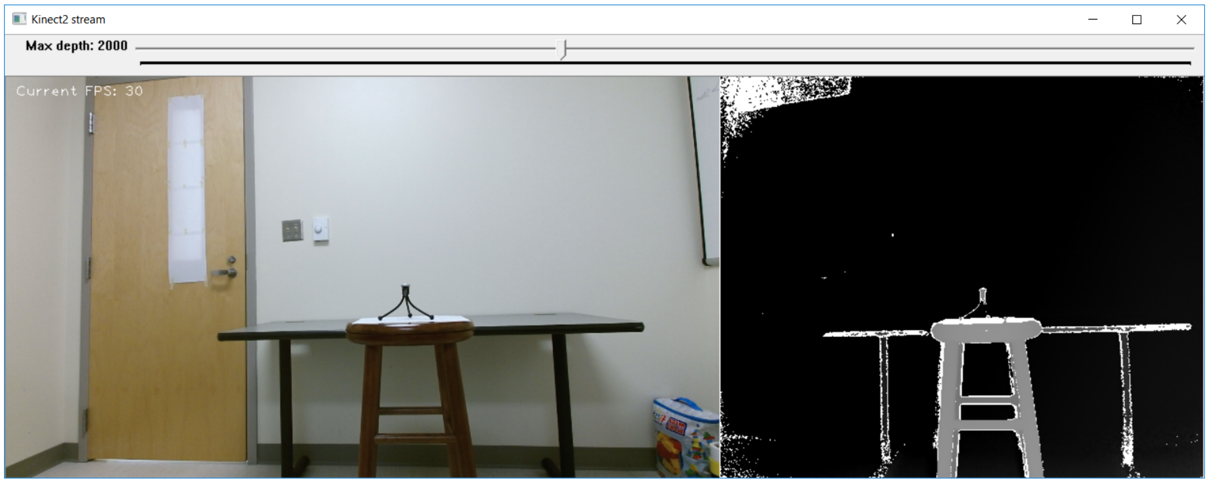


Figure 5.1: Background scene with corresponding depth image

We obtain 50 frames of each view from a static camera with the corresponding depth matrices stored in a MongoDB database. Averaging these frames gives us a smoother image with null values removed as shown in Figure 5.3. A simple absolute pixel

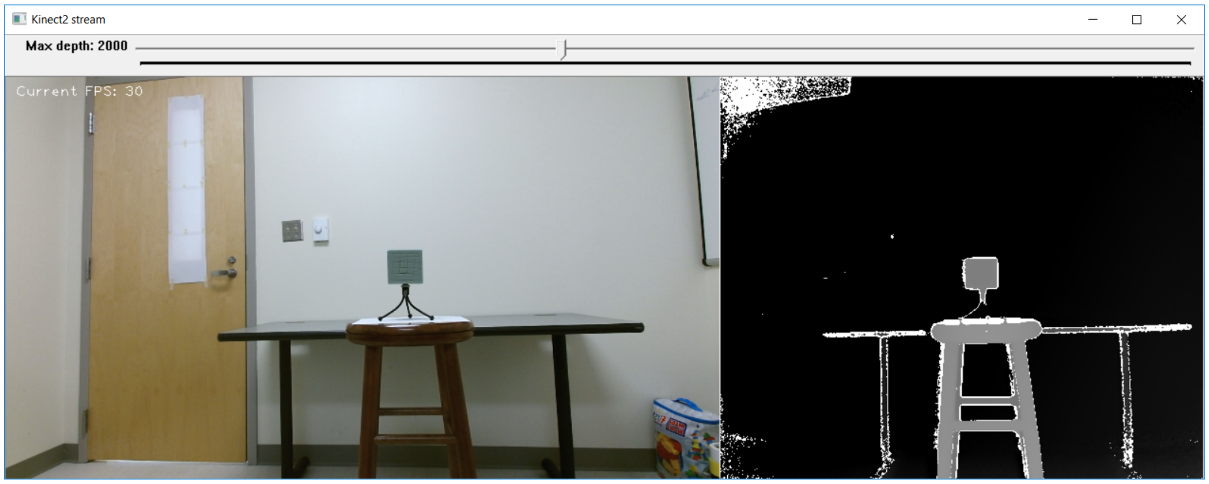


Figure 5.2: Scene with object added in with corresponding depth image

subtraction is performed outputting the image Figure 5.4 that we will be using in Canny Edge Detection (Section 5.1.1).

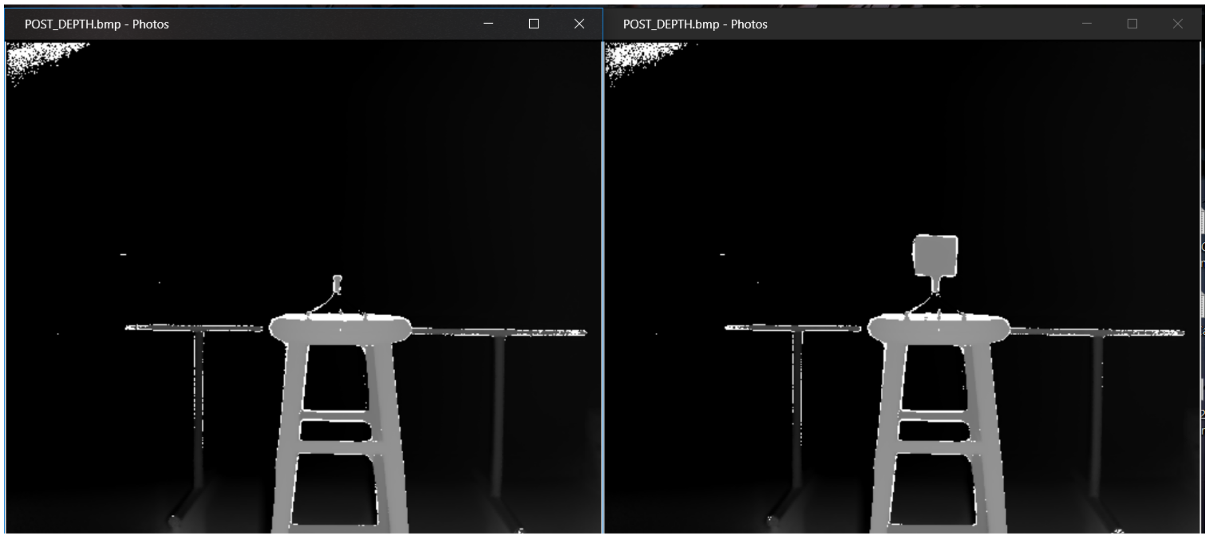


Figure 5.3: Averaged scene of background and object frames

5.1.1 Canny Edge Detection and Size Calculation

The final pixel subtracted image can be used now that all distracting background features and noise is removed. Canny edge detection is performed on the image as shown in Figure 5.5. The minimum threshold slider at the top can be used to adjust the number of edges shown. Finding a good threshold varies with the object as only the outer-most outline of the object is needed. The pixel and corresponding depth value can be selected from two points along the detected edge. If an edge is not selected, the closest pixel along the edge will be selected instead. The real world distance between the two points selected can be found using the methods described in Section 3.4. It was found that selecting two points along the same plane resulted in the most accurate measurement. The dimension calculated will be used to scale the 3D model to the correct size in the 3D Builder app.

5.2 Point Cloud Generation

Using VisualSFM, we open the image sequence for the object and perform the image matching stage. Afterwards a sparse and dense point cloud of the object scene is generated from the image sequence using SIFT and CMVS. This results in a noisy point cloud as shown in Figure 5.6. In Meshlab, we import this point cloud and clean up any stray artifacts that may affect the surface reconstruction as shown in Figure 5.7.

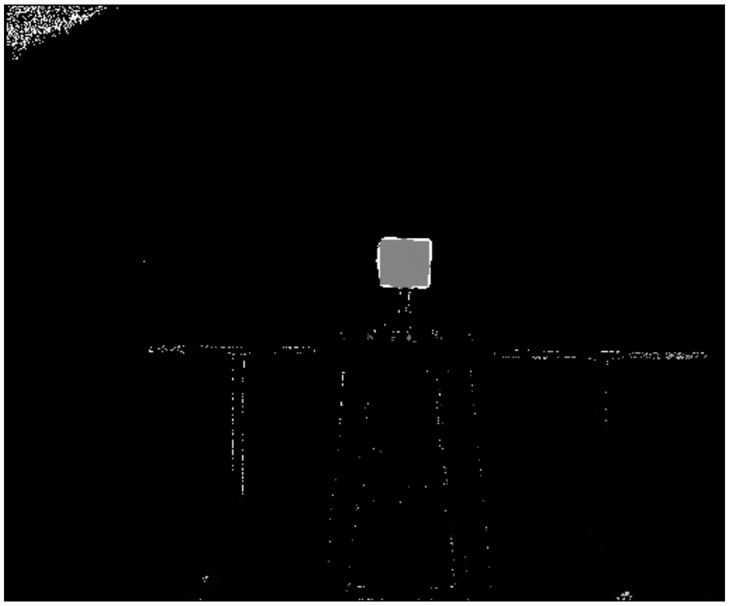
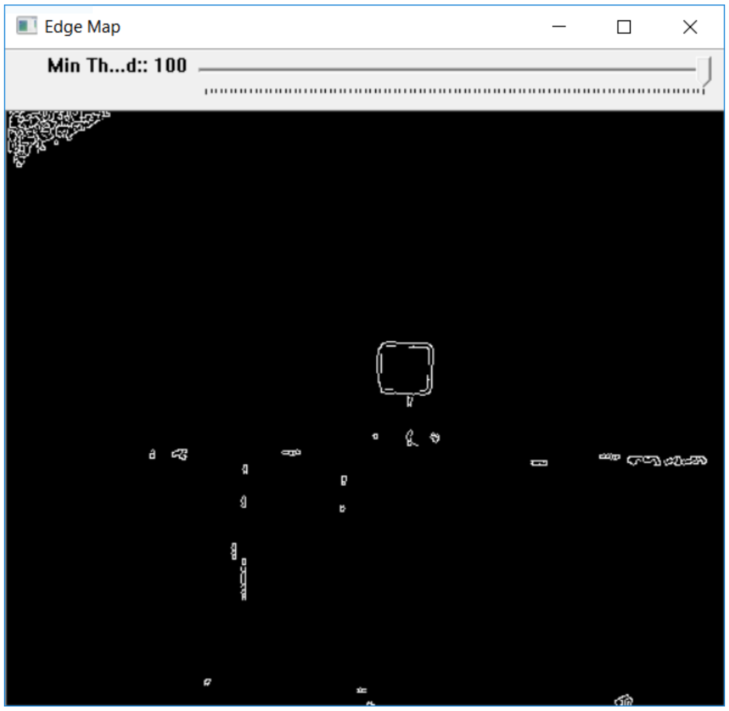
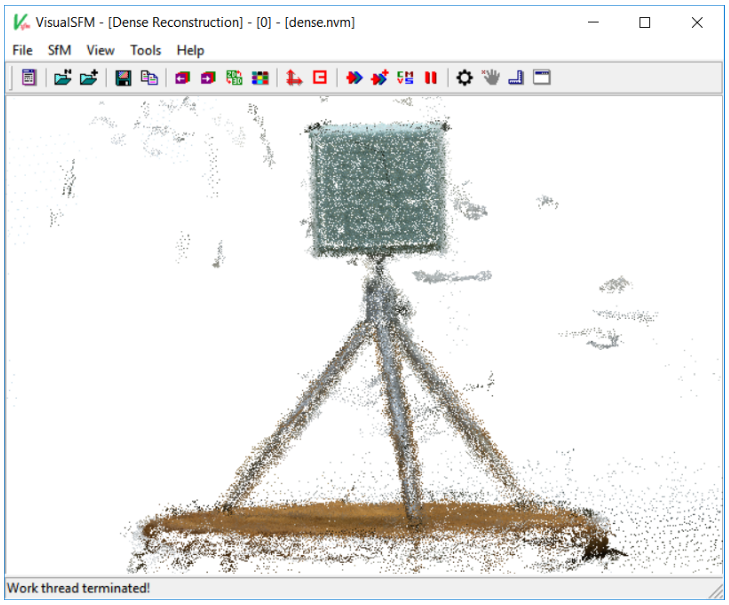


Figure 5.4: Result after applying the pixel subtraction to the averaged background and object frames


Edge Map

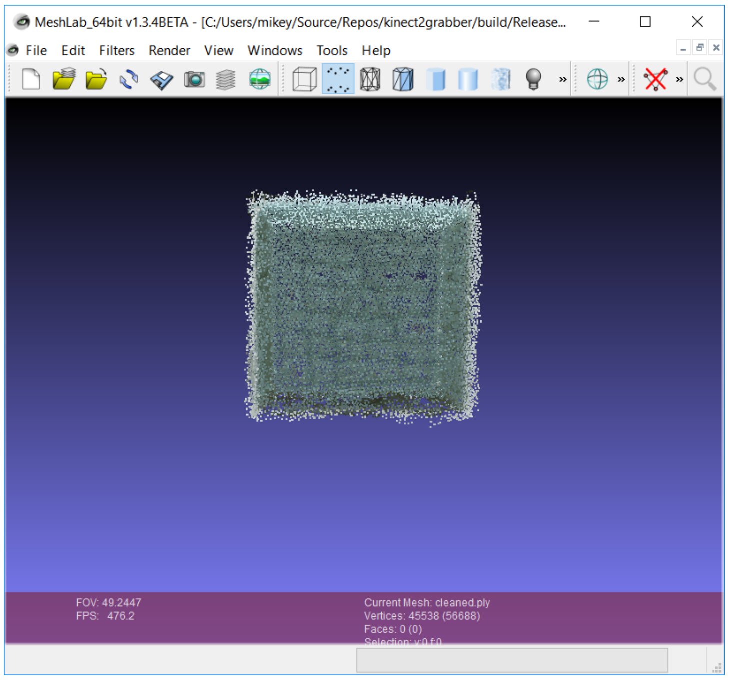
Min Th...d:: 100

Figure 5.5: Canny edge detected image of the object frame of a cube object with minimum threshold set to 100



Work thread terminated!

Figure 5.6: Dense point cloud of cube object on tripod using VisualSFM

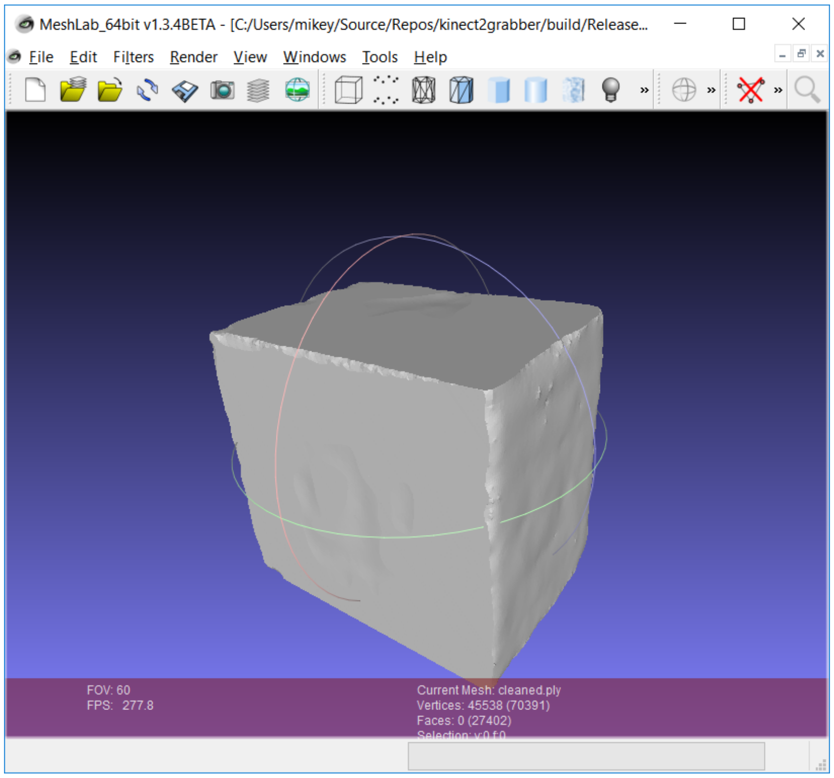

MeshLab_64bit v1.3.4BETA - [C:/Users/mikey/Source/Repos/kinect2grabber/build/Release...
File Edit Filters Render View Windows Tools Help

FOV: 49.2447
FPS: 476.2
Current Mesh: cleaned.ply
Vertices: 45538 (56688)
Faces: 0 (0)
Selection: v:0 f:0

Figure 5.7: Cleaned dense point cloud of cube object using Meshlab

5.3 Surface Reconstruction

Once we have cleaned up the dense point cloud in Meshlab we can perform surface reconstruction using the Poisson surface reconstruction algorithm described in Section 3.5.2 without fear of any noise that would affect the accuracy of the mesh. Different values for each of the parameters for the reconstruction algorithm are tested and the best results are found to be low to mid values for "octree depth" and "solver divide," high values for "samples per node," while keeping "surface offset" at a value of 1. Choosing good values for these parameters produces the model shown in Figure 5.8.


MeshLab_64bit v1.3.4BETA - [C:/Users/mikey/Source/Repos/kinect2grabber/build/Release...
File
Edit
Filters
Render
View
Windows
Tools
Help

```
FOV: 60
FPS:  277.8

Current Mesh: cleaned.ply
Vertices: 45538 (70391)
Faces: 0 (27402)
Selection: v:0 f:0
```

Figure 5.8: Poisson surface reconstruction model of cube object

Meshlab saves the model as an .obj file and does not retain any information about the size of the object. The model is saved and imported into the 3D Builder Windows app. Using the measurement found in Section 5.1.1 we can scale it to the correct dimension and export it as an .stl file, a file type that retains information about the size of the model.

5.4 3D Model Print

Once the model has been scaled to size, it is ready for printing. 3D printing takes time proportional to the number of triangles used to make the mesh; the more detailed the object, the longer it will take to print. The software that is used to 3D print can further reduce the number of triangles used in the reconstruction, simplifying the model and reducing the time it will take to print.

Table 5.1 shows the comparison of dimensional values between the original object and the first 3D printed model. The model was corrected using the methods in Section 5.5 and printed again. Table 5.2 shows the comparison of dimensional values between the original object and the corrected model. At most a dimension was off by 1.11% which is a significant improvement from the first model. Figure 5.9 shows comparisons between the original object and the corrected 3D printed model from different angles. Some important features to note are the vertices and edges where the model is uneven. Although not clearly visible from Figure 5.9, the bottom of the model is not as uniformly flat as the other faces. The larger percent error of the model's height and the non-uniformity is due to the image capturing stage. The object was resting on a raised stand and the area that was in contact with the model produced a more concentrated area of points in the point cloud generation thus influencing the weight of each point during the surface reconstruction.

Table 5.1: Value comparison between original cube and first 3D printed model

	Original	First Model	Difference	Percent Error
Length (cm)	8.70	8.83	0.13	1.49
Width (cm)	6.70	6.45	0.25	3.73
Height (cm)	9.0	8.7	0.3	3.33

Table 5.2: Value comparison between original cube and corrected 3D printed model

	Original	Corrected Model	Difference	Percent Error
Length (cm)	8.70	8.65	0.05	0.57
Width (cm)	6.70	6.65	0.05	0.75
Height (cm)	9.0	8.9	0.1	1.11

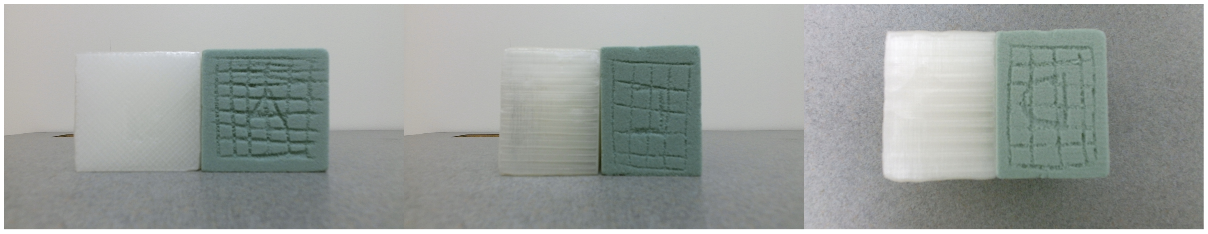


Figure 5.9: Comparisons of the 3D printed model and original cube

This thesis aims to be able to scale a model to the size of any object of the size within the effective view of the Kinect and thus a model of a stress ball was also 3D printed following the same process as the cube. Multiple diameters of the ball were measured and averaged. The same was done for the model. The average diameter of the ball and the first model are listed in Table 5.3. The model was corrected using the methods in Section 5.5 and printed again. Table 5.4 shows the comparison of dimensional values between the original object and the corrected model. The corrected model is compared with the original in Figure 5.10. The diameter of the model is off by a larger percentage than the cube. This is to be expected as the ball is of a smaller size than the cube. As the size of the scanned object decreases, the inherent uncertainty of the Kinect's depth sensor accuracy begins to have more weight. As with the cube, some features that are not immediately visible from Figure 5.10 are the irregularities along the bottom of the model in the form of small bulges. The rest of the model is uniformly smooth due to the lines and texture of the original ball which allows for more distinct points to be matched and helps create a more precise point cloud.

Table 5.3: Value comparison between original ball and first 3D printed model

	Original	First Model	Difference	Percent Error
Average Diameter (cm)	6.20	6.46	0.26	4.19

Table 5.4: Value comparison between original ball and corrected 3D printed model

	Original	Corrected Model	Difference	Percent Error
Average Diameter (cm)	6.20	6.00	0.20	3.22

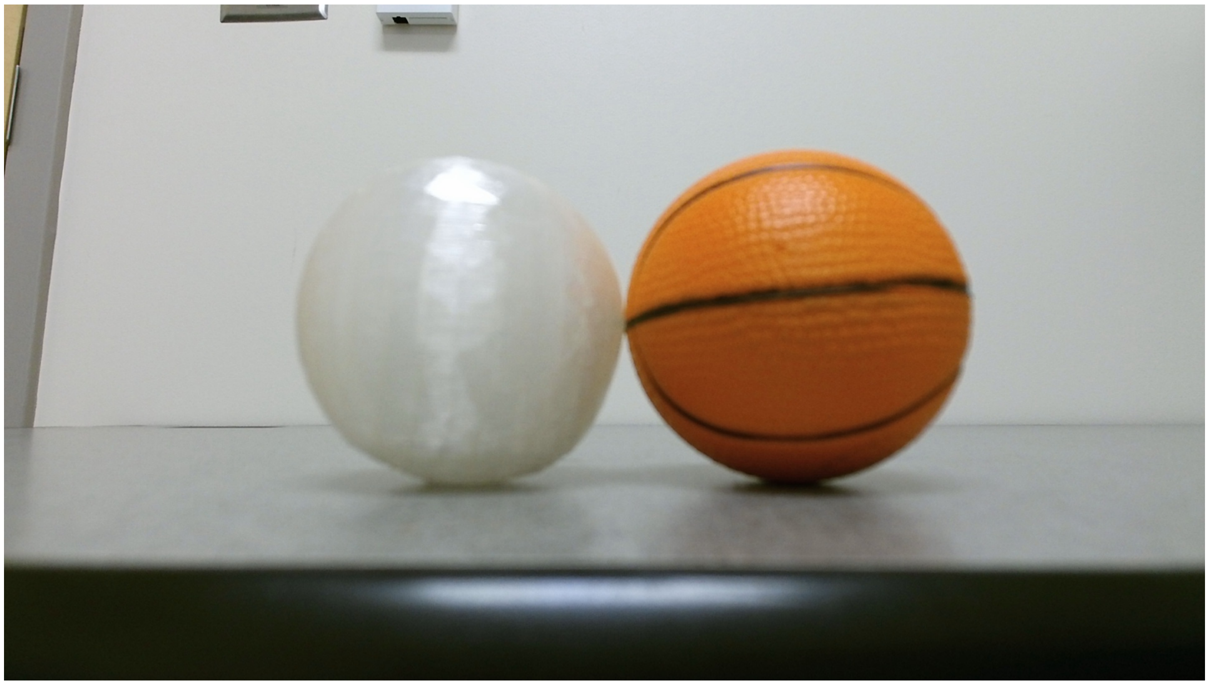


Figure 5.10: Comparison of 3D printed model and original ball

5.5 Error Correction

The initial results were not acceptable as they didn't match the dimensions of the objects and were misshapen. Corrections were made resulting in the models shown in Figure 5.11 and Figure 5.12. It was found that objects with low textures or few discernible features produced point clouds that lacked any data points in such areas except for the edges. For example, the cube used in this thesis was originally a solid green floral foam cube with flat sides. Only a wire frame of the edges was produced when point clouds were generated from the unmodified cube. To correct this, a grid was etched into each side to give the cube more texture and unique points to be matched.

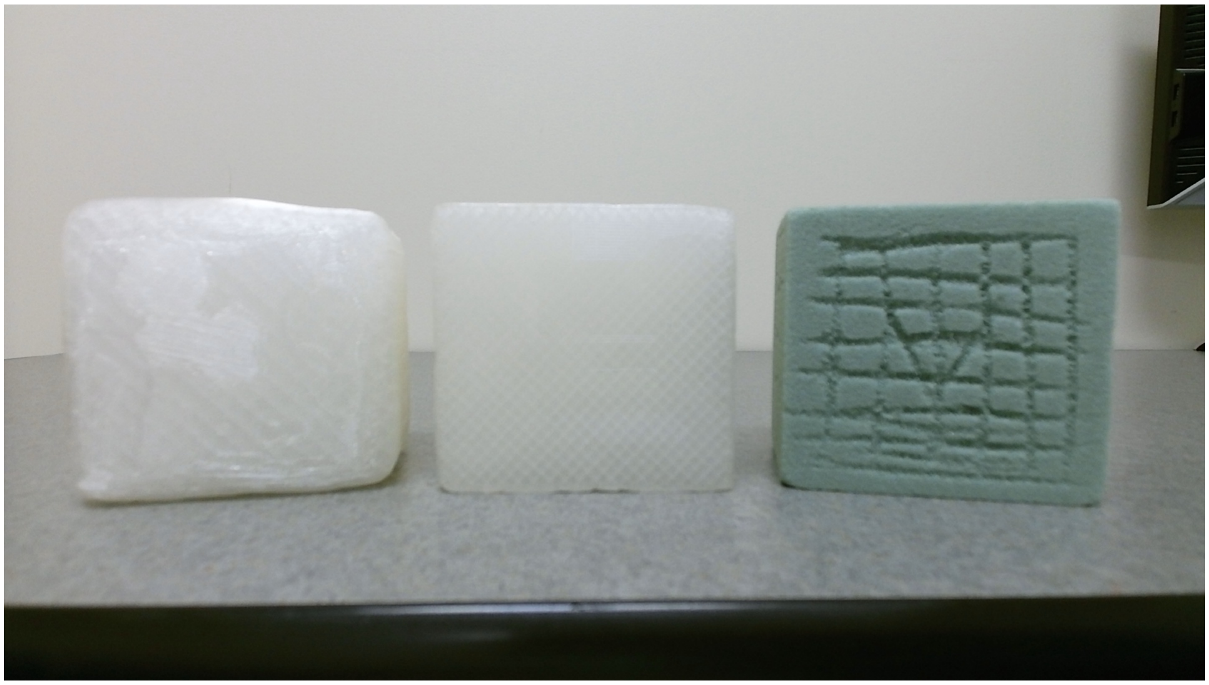


Figure 5.11: From left to right: the first model, corrected model, and the original cube

In the edge detection stage, sufficient noise in the depth image could result in no depth measurement read at a specific pixel along the edge. This was addressed by iteratively stepping inward from the selected pixel until a pixel with a depth measurement was found. In objects that have rounded edges or edges that are slanted or otherwise at a different depth, the usefulness of this solution can deteriorate since each pixel represents a point a certain number of millimeters away from the selected pixel depending on the distance from the sensor. As discussed in Chapter 2, different colors, reflectivity, and clearness of the material can influence the depth sensing. An opaque object that is neither exceptionally dark or light should be chosen.

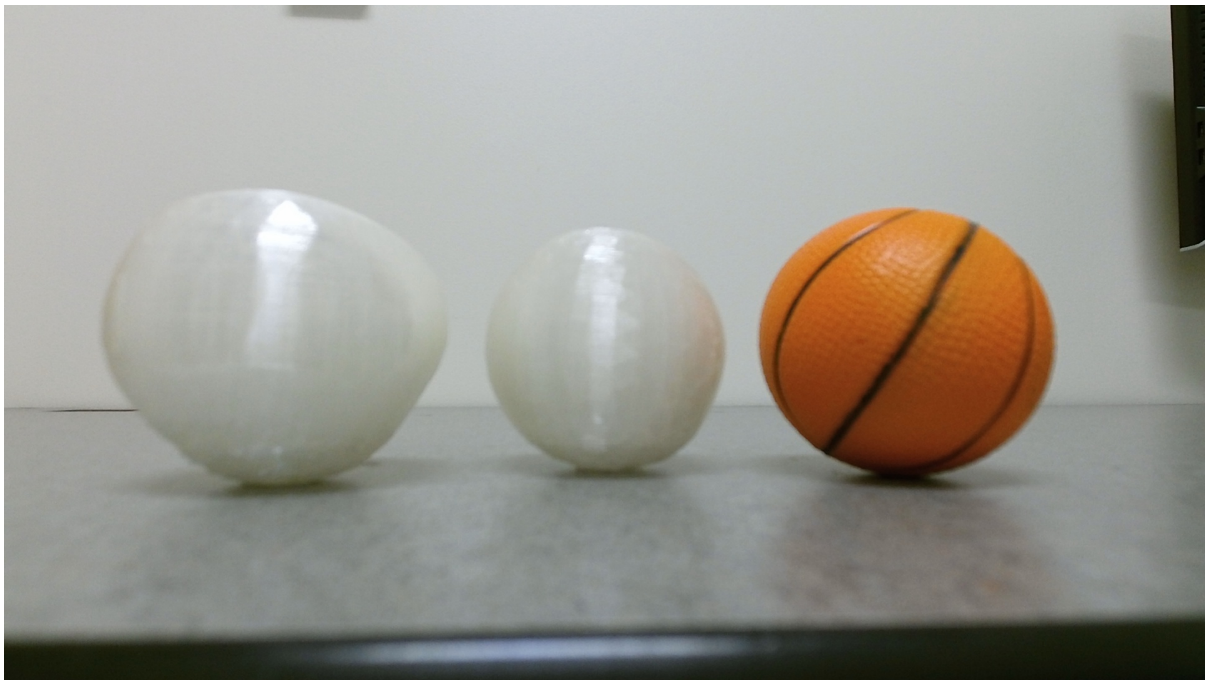


Figure 5.12: From left to right: the first model, corrected model, and the original ball

Chapter 6 - Summary and Conclusion

6.1 Background Subtraction and Edge Detection

The background subtraction used in this thesis was a simple pixel subtracting algorithm. This is an important stage in filtering out noise for edge detection as shown in Figure 5.5. Each depth frame returned by the Kinect produces slight deviations in the measured depth, sometimes no depth at all, at each pixel. Averaging the values at each pixel and subtracting the background produces a sufficient frame that can be analyzed.

Canny edge detection produces a clean outline of the object and can be further cleaned by adjusting the minimum threshold. The edge pixels that are selected may not have a measured depth associated with it and an algorithm to find the next closest pixel with a depth is implemented. This solution begins to fail if the object does not have clear edges as seen by the larger error of the ball.

The Kinect sensor has an effective range between 0.5m - 4m and a resolution of 512 x 424. For small objects at a distance of 0.5m away from the sensor, the depth becomes more difficult to capture around the edges.

6.2 Point Cloud Generation

Point cloud generation was implemented using SIFT and CMVS to match unique features between sequential images and to build a point cloud from different views. Images were captured by moving a Kinect around a static object and using discernible features around the room to build a point cloud in VisualSFM. The object may also be placed on a rotating platform with a static Kinect. The object must have enough discernible features for the algorithm to produce a complete cloud.

A concentration of points is apparent around the area of contact between the object and the platform. As there is no way of suspending an object in mid-air while scanning the bottom of the object, it is difficult to clean this concentration of points. A possible solution would be using a rotating platform so long as the platform and background has no discernible feature. The object placed on this platform can then be orientated differently to simulate being suspended in mid-air. For small objects without many features, it is difficult for this method to produce a complete model.

6.3 3D Model and 3D Print

A 3D model was generated using Poisson surface reconstruction. The reconstruction is reliant upon a sufficiently detailed point cloud and good parameters chosen for the octree depth, solver divide, samples per node, and surface offsetting. A 3D printed model of a cube was produced with a percent error of 1.11% in the height. The Kinect depth sensor has a standard deviation of 10mm at a range of 0.5m which can account for some of the error in the dimensions. The model of the ball featured a percent error of 4.304% in the diameter which the Kinect depth sensor standard deviation cannot fully account for, however, the edge detection stage can account for this increase in error due to the difficulty in gathering the depth information around the edges of sufficiently small objects.

6.4 Future Work

The process pipeline using a Kinect, VisualSFM, and Meshlab presented in this thesis offers a cheap alternative to 3D scanning objects relative to 3D laser scanners. However, it does suffer from some drawbacks. Objects that are made of dark, reflective, or clear material poses problems for scanning and measuring the dimensions of said object. At each stage, there are improvements that can be made to reduce the error of the model. Future research can begin exploring ways to address material composition as well as further minimizing sources of error.

Bibliography

[1] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence 6: 679-698*, 1986.

[2] Yasutaka Furukawa. Clustering views for multi-view stereo, 2011.

[3] K. Khoshelham. Accuracy analysis of kinect depth data. *ISPRS Workshop Laser Scanning 38(5): W12*, 2011.

[4] L. Li. Time-of-flight camera - an introduction. Technical report, Texas Instruments, May 2014.

[5] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision, 30, 2: 117-154*, 1998.

[6] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60(2): 91–110*, 2004.

[7] Andersen, M., Jense, T., Lisouski, P., Mortensen, A., Hansen, M. et al. Kinect depth sensor evaluation for computer vision applications. Technical report, AARHUS University Electrical and Computer Engineering, 2015.

[8] Barrow, H.G., Tenenbaum, J.M. Interpreting line drawings as three-dimensional surfaces. *Artificial Intelligence vol. 17 issues 1-3: 75-116*, 1981.

[9] Bueno, M., Diaz-Vilarino, L., Martinez-Sanchez, J., Gonzalez-Jorge, H. et al. Metrological evaluation of kinectfusion and its comparison with microsoft kinect sensor. *Measurement 73: 137-145*, 2015.

[10] Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganoveli, F., Ranzug, G. Meshlab: an open-source mesh processing tool. In *Proceedings of Eurographics Italian Chapter Conference*, Piza, Italy, 2008.

[11] Corti, A., Giancola, S., Mainetti, G., Sala, R. A metrological characterization of the kinect v2 time-of-flight camera. *Robotics and Autonomous Systems vol. 75: 584-594*, 2016.

[12] Gonzalez-Jorge, H., Rodriguez-Gonzalvez, P., Martinez-Sanchez, J. et al. Metrological comparison between kinect i and kinect ii sensors. *Measurement 70: 21-26*, 2015.

[13] Han, J., Shao, L., Xu, D., Shotton, J. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE Transactions on Cybernetics 43(5): 1318-1334*, 2013.

[14] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P. et al. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on user interface software and technology 559-568*, Santa Barbara, CA, 2011.

[15] Jaiswal, M., Xie, J., Sun, M. 3d object modeling with a kinect camera. In *Proceedings of Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, Chiang Mai, Thailand, 2014.

[16] Kazhdan, M., Bolitho, M., Hoppe, H. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing 61-70*, Cagliari, Sardinia, Italy, 2006.

[17] Lachat, E., Macher, H., Landes, T., Grussenmeyer, P. Assessment and calibration of a rgb-d camera (kinect v2 sensor) towards a potential use for close-range 3d modeling. *Remote Sensing 7(10): 13070-13097*, 2015.

[18] Lachat, E., Macher, H., Millet, M., Landes, T., and Grussenmeyer, P. First experiences with kinect v2 sensor for close range 3d modelling. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences vol. 40 no. 5: 93-100*, 2015.

[19] Lembke, E., Gordet, J. Reconstruction of 3d objects from images to render and print to scale. In *Proceedings of Research Experience for Teachers at Appalachian State University*, Boone, NC, 2015.

[20] Liu, X., Yang, M., Wang, C., Yang, G. Approach for 3d reconstructing bonsai plant. In *IEEE International Conference on Multimedia Big Data (BigMM) 366-370*, 2015.

[21] Manikhi, O., Adlkhast, B. A 3D object scanner: An approach using microsoft kinect. master's thesis in Information Technology 120 ECTS. Halmstad University, Halmstad, Sweden, 2013.

[22] Moons, T., Van Gool, L., Vergauwen, M. *3D reconstruction from multiple images, part 1: Principles.* Now Publishers Inc., 2009.

[23] Newcombe, R., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. et al. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of IEEE international symposium on mixed and augmented reality (ISMAR) 127-136*, Basel, Switzerland, 2011.

[24] Pagliari, D., and Pinto, L. Calibration of kinect for xbox one and comparison between the two generations of microsoft sensors. *Sensors 15: 27569-27589*, 2015.

[25] Rusinkiewicz, S., Hall-Holt, O., Levoy, M. Real-time 3d model acquisition. *ACM Tansactions on Graphics (TOG) vol. 21 no. 3: 438-446*, 2002.

[26] Santoso, F., Garratt, M., Pickering M., Asikuzzaman, M. 3d mapping for visualization of rigid structures: A review and comparative study. *IEEE Sensors Journal 16(6): 1484-1507*, 2016.

[27] Sarbolandi, H., Lefloch, D., Kolb, A. Kinect range sensing: Structured-light versus time-of-flight kinect. *Computer Vision and Image Understanding 139: 1-20*, 2015.

[28] Steward, J., Lichti, D., Chow, J., Ferber, R., Osis, S. Performance assessment and calibration of the kinect 2.0 time-of-flight range camera for use in motion capture applications. *FIG Working Week 1-14*, 2015.

[29] Tsai, F., Wu, T., Lee, I., Change, H., Su, A. Reconstruction of indoor models using point clouds generated from single-lens reflex camera and depth images. *The International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences 40(4): 99-102*, 2015.

[30] Wu, C., Agarwal, S., Curless, B., Seitz, S. Multicore bundle adjustment. In *CVPR*, 2011.

[31] Xu, H., Wang, X., Shi, L. Fast 3d-object modeling with kinect and rotation platform. In *Proceedings of 2015 Third International Conference on Robot, Vision and Signal Processing (RVSP) 43-46*, 2015.

[32] Yang, L., Zhang, L., Dong, H., Alelaiwi, A., El Saddik, A. Evaluating and improving the depth accuracy of kinect for windows v2. *IEEE Sensors Journal 15(8): 4275-4285*, 2015.

[33] C. Wu. Siftgpu: A gpu implementation of scale invariant feature transform (sift). http://cs.unc.edu/ ccwu/siftgpu, 2007.

[34] C. Wu. Visualsfm: A visual structure from motion system. http://ccwu.me/vsfm, 2011.

Chapter A - Appendix

The source code and programs used in this thesis are included on the enclosed CD. The following is a list of files that appear on this CD:

1. Kinect2 Grabber.sln: Contains source code for the program used to capture and record RGB images and depth data from Microsoft Kinect.
2. BackgroundSubtractor.sln: Contains source code for the program used for background subtraction.
3. EdgeDetector.sln: Contains source code for the program using Canny edge detection for size calculation.
4. process.py: Contains source code for storing object depth information in MongoDB and averaging frames.
5. VisualSFM: Free software used to create point clouds of scanned objects.
6. Meshlab: Free software used to generate 3D mesh using Poisson surface reconstruction.
7. 3D Builder: Free software provided by Microsoft for editing 3D object meshes.

Vita

Michael Yang was born in 1993, in Chapel Hill, North Carolina to Bing Yang and Mei Hu. Accepted to Appalachian State University in 2011, he began studying Physics, but switched to Computer Science after taking his first Computer Science class and discovering his passion for programming. He earned his Bachelor of Science in Computer Science in 2016 and proceeded into graduate studies also at Appalachian State University through the Accelerated Admission program, graduating in the summer of 2017.