TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

Understanding '3D Reconstruction'

Report prepared by:

Aasman Bashyal (074BEX402)
Basanta Rijal (074BEX409)
Pawan Sharma Poudel(074BEX419)
Saju Khakurel (074BEX438)

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING,
CENTRAL CAMPUS, IOE, LALITPUR, NEPAL

March, 2020

Acknowledgement

Completing the Computer Graphics Project is harder but interesting than we thought, and more rewarding than we could have ever imagined. Most special thanks to our respected Lecturer Er. Basanta Joshi for his guidance at each steps of our journey. We would as well like to acknowledge the Department of Electronics and Computer Engineering for providing us an opportunity of Graphics Project as a part of our course of Computer Graphics.

Lastly, a very special thanks to our parents, co-workers, seninors, classmates and all the supporting hands and minds who wished us good luck for the successful completion of our project.

Abstract

This report reviews the wide accepted pipeline of 3D reconstruction as: data acquisition and refinement point cloud generation, alignment and merging and then surface reconstruction. Common practices of these steps require well sophisticated knowledge of the mathematics, computer vision, optimization, KDtree classification and so on. ICP, the most widely used algorithm for Point Cloud alignment requires a preliminary knowledge of correspondence matching (KDTree) and optimization (Differential Counter Bound, ARGMIN/ARGMAX. RMSE). The major purpose of this report is to devise a simple pipeline for complete 3D point cloud generation that provides a generic overview of 3D RECONSTRUCTION for easiest understanding. This simplification comes at a cost of optimization and quality of output.

First of all, this report uses the simplest averaging method for input data refinement. Secondly, it bypasses the camera pose estimation using ICP for point cloud alignment by mechanically constraining and identifying camera poses. Thus, the required knowledge is reduced to basic arithmetic, linear algebra and a bit of pinhole camera model. The results show that the point clouds are not perfectly aligned as expected. This could be a result of the sensor error; camera pose error and mechanical uncertainty. This method is not recommended for research or practical purposes due to its inferiority in output, however it can be very useful for a basic understanding of the generic procedure of 3d reconstruction.

Table of Contents

List Of Figures

1 Introduction

'3D Reconstruction' is the process of creation of 3D models of real-life objects in fields such as 'Computer Vision' and 'Computer Graphics'. 3D Reconstruction includes 3D data acquisition and surface reconstruction. 3D data acquisition is mainly done in two ways.

a. Passive Methods
b. Active Methods

Passive Methods make use of the radiance reflected or emitted by the object of interest itself instead of artificial illumination from the sensor being used. The construction of the 3D structure is based on image understanding. Cameras are the typical sensors used in the process and '*Photogrammetry*' is the typical method. It can be achieved from *Multiview approaches* in which multi view geometry is used for camera localization and 3D point estimation on images of target from more than one viewpoint either using stereo camera systems or a single camera in motion over a period of time (structure from motion). Alternatively, *Single view approaches* infer 3D shape from a single viewpoint depending on information cues like shading, texture, focus for 3D modelling.

Active Methods, in contrast to passive methods, emit artificial illumination or other forms of electromagnetic radiation or may even use mechanical methods to get the depth map of the object of interest. The depth map along with other sensor information (typically RGB-D) are used to generate point cloud of the target. One of the techniques used in active methods is *Structured Light* in which distortions in sequence of known patterns of E.M. radiation projected, due to the geometry of the target, are observed and analysed to generate depth maps. This technique is often referred to as *Active Stereo Vision* because of its similarity with *Binocular Stereo Vision* in passive methods. Time of Flight is another technique in which depth map is generated based on the time taken by radiation emitted from source to reflect from the

surface of the targets. *Microsoft Kinect* is an example of an active 3D imaging system.

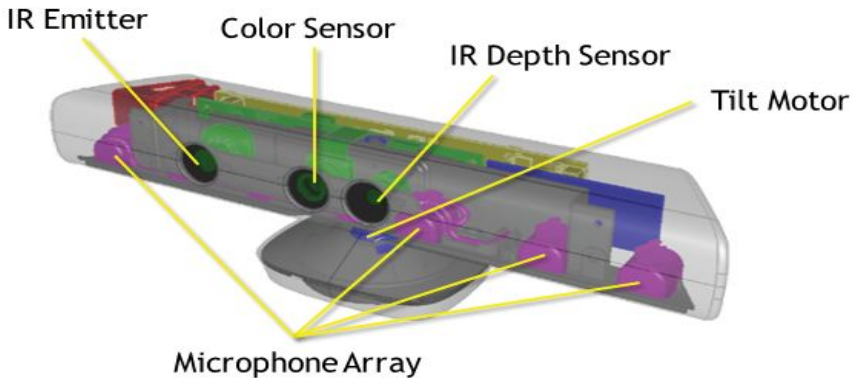


Figure 1 Kinect's Internal Sensors Visualization

1.1 Problem Statement

3D reconstruction of the real-life objects is done using Kinectv1 that sends the specific pattern of ir rays to the surface to build. It draws the point cloud of various points on the object based on the time taken by the striking rays to reflect back to the receiver. The depth map along with other sensor information (typically RGB-D) are used to generate point cloud of the target. Then, translation, manipulation of co-ordinates, alignment and merging of the points is done in processing3 to finally generate the 3D model of the object.

1.2 Objective

- Simplify 3D Reconstruction Process for better understanding.
- The major objective of this project is to help us understand the general overview behind the idea of 3D reconstruction.
- To create Point Cloud representation of a real life 3D object.

1.3 Scope

3D reconstruction is one of the basic elements of Computer Vision and Computer Graphics hence it finds its applications among many different fields involving CAD such as:

- Pavement engineering
- Medicine
- Free-Viewpoint video reconstruction
- Robotic mapping
- City planning
- Tomographic reconstruction
- Gaming
- Virtual environments and virtual tourism
- Earth observation
- Archaeology
- Augmented reality
- Reverse engineering
- Motion capture
- 3D object recognition, gesture recognition and hand tracking

Source: Wikipedia

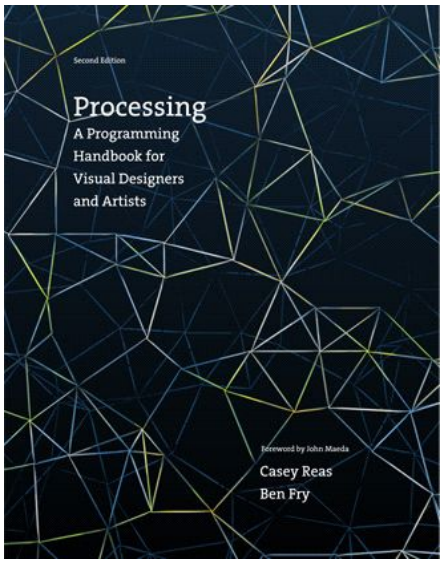
This project can be used as core idea for different applications that can be developed as extensions which are listed as:

- Simultaneous Localization and Mapping (SLAM) from point cloud representation
- Digitalization of environments and cities using photogrammetry.
- Virtual tour systems of local tourist attractions
- Augmented descriptions of local tourist attractions
- Augmented projections of past states of a 3D environment like monuments, temples

2 Background Theory

2.1 Processing

Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. It uses the Java language which is default language for this software, with additional simplifications such as additional classes and aliased mathematical functions and operations. It also provides a graphical user interface for simplifying the compilation and execution stage. Everything that Processing draws on the screen with the P2D and P3D renderers is the output of an appropriate "default shader" running behind the scenes. Processing handles these default shaders transparently so that the user doesn't need to worry about them. a shader is basically a program that runs on the Graphics Processing Unit (GPU) of the computer, and generates the visual output we see on the screen given the information that defines a 2D or 3D scene.



2.2 KinectV1

Kinect is a line of motion sensing input devices that provides a natural user interface (NUI) that allows users to interact intuitively and without any intermediary device, such as a controller. The camera detects the red, green, and blue color components as well as body-type and facial features. It has a pixel resolution of 640x480 and a frame rate of 30 fps. This helps in facial recognition and body recognition. The depth sensor contains a monochrome CMOS sensor and infrared projector that help create the 3D imagery throughout the room. It also measures the distance of each point of the player's body by transmitting invisible near-infrared light and measuring its "time of flight" after it reflects off the objects.

Figure 3Kinect

2.3 3D Reconstruction

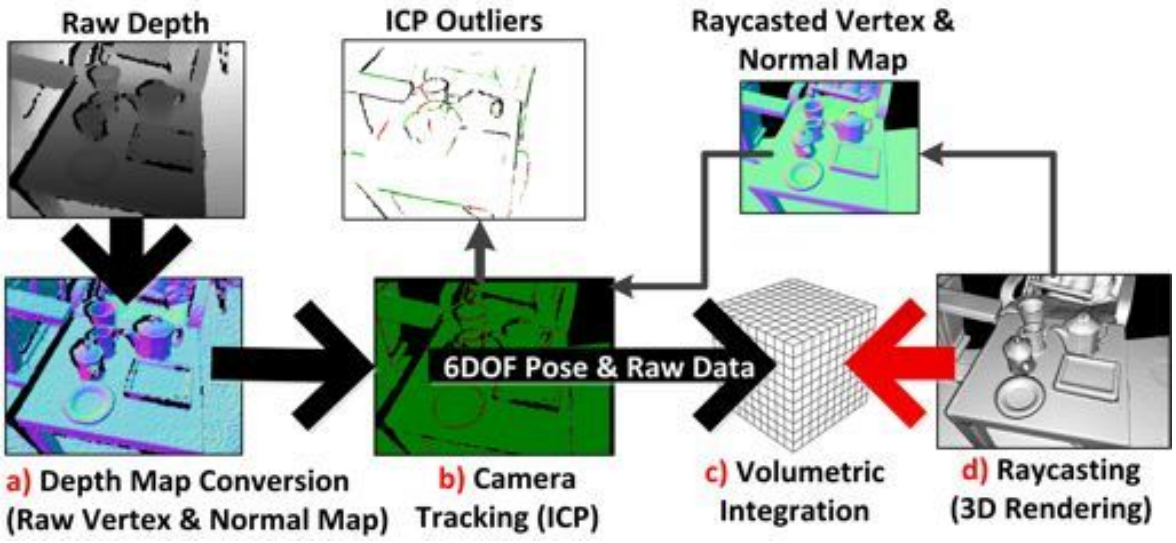


Figure 4 3D Reconstruction pipeline

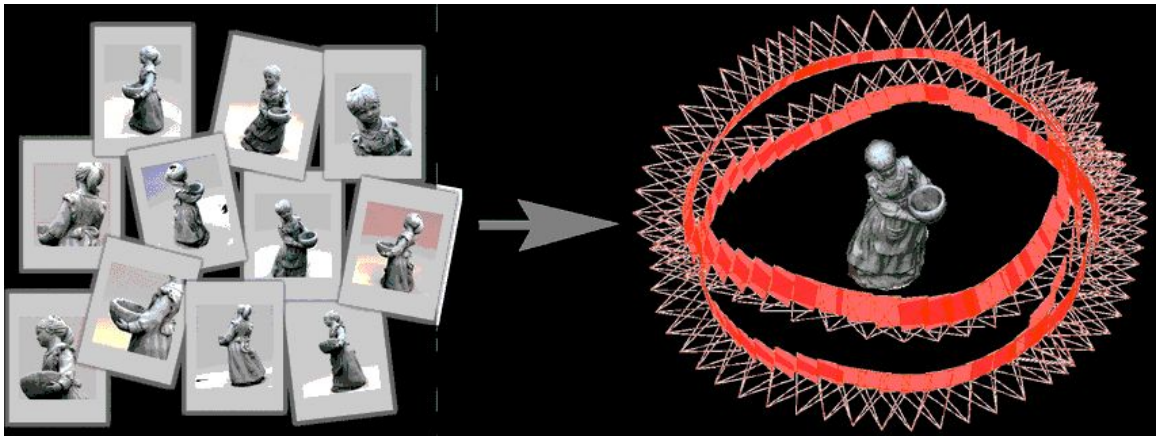


Figure 53D Reconstructed model

2.4 Open CV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. In simple language it is library used for Image Processing. It is mainly used to do all the operation related to Images. OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface.

2.5 Camera Calibration

Camera sectioning is the process of estimating the parameters of a pinhole camera model approximating the camera that produced a given photograph or video. Usually, the pinhole camera parameters are represented in a 3 × 4 matrix called the camera matrix. Simply it is the determination of the relationship between the 3D position of a

point in the world and the 2D pixel coordinates of its image in the camera. In this project, we explore extension of one algorithm for calibrating a single camera to calibrating an array of 128 cameras. Our primary goal is implementing a global, nonlinear optimization procedure to compute "optimal" values of all camera parameters. We hope to achieve more accuracy and stability this way, as opposed to using existing software to calibrate each camera separately.

2.6 Point Cloud

A point cloud is a set of data points in space. Point clouds are generally produced by 3D scanners, which measure many points on the external surfaces of objects around them. They are aligned with 3D models or with other point clouds, a process known as point set registration. Point clouds are commonly generated using 3D laser scanners and LIDAR technology and techniques. Here, each point represents a single laser scan measurement.

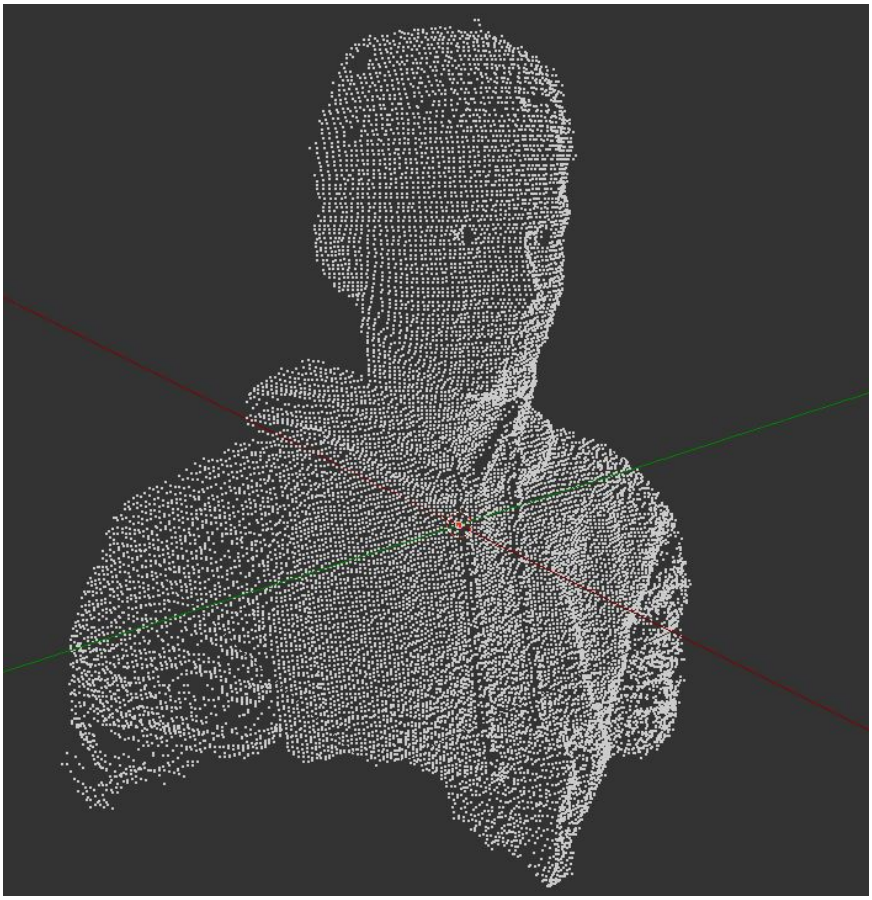


Figure 6Point Cloud Object

2.7 Camera Parameters

Camera parameters are the parameters used in a camera model to describe the mathematical relationship between the 3D coordinates of a point in the scene from which the light comes from and the 2D coordinates of its projection onto the image plane. The intrinsic parameters, also known as internal parameters, are the parameters intrinsic to the camera itself, such as the focal length and lens distortion. The extrinsic parameters, also known as external parameters or camera pose, are the parameters used to describe the transformation between the camera and its external world.

3 Methodology

To reduce the complexity to minimum, Microsoft Kinect RGB-D sensor is chosen as the depth sensor. An experimental setup to take the input RGBD frames by rotating the object of interested was constructed. Because of ease in programming, efficient real time rendering and a pre-existing Kinect library PROCESSING IDE was selected as the programming interface.

3.1 System Block Diagram

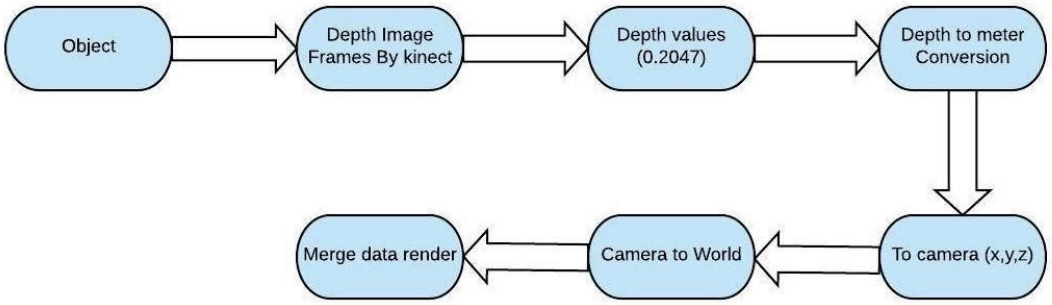


Figure 7System Block Diagram

3.1.1 Depth Image from Kinect:

Kinect is placed stationary at a point 0.6 meters from the centre of the object of interest, placed on a rotating platform. The depth image frame is taken by only rotating the object by a known angle about the plane parallel to Y- axis of Kinect and passing through the centre of the object. The depth frame is refined by taking an average of multiple frames from the same position. A virtual bounding box is created using threshold values for clipping unnecessary points.

3.1.2 Depth to Meters Conversion:

The depth values obtained lie in a range [0-2047] which need to be converted to meters for giving meaning to them. For this, the function from depth calibration from

Kinect node is used, which appears to be linear and was determined as best fit for experimental data.

F(x) = (1.0 / (x * -0.0030711016 + 3.3309495161))

3.1.3 Image Coordinate to Camera Coordinate Conversion:

Though the z coordinate is obtained as depth value, the x and y coordinates are still pixel coordinates and need to be converted to actual x and y coordinates on the Kinect coordinate. This is obtained using the intrinsic camera as:

z = meter depth
x = (u- cx_d) * z* fx_d)
y = (v - cy_d) * z* fy_d)

where: fx_d and fy_d are reciprocal of focal lengths of the depth camera, cx_d and cy_d are the principal points of depth camera (center of image)

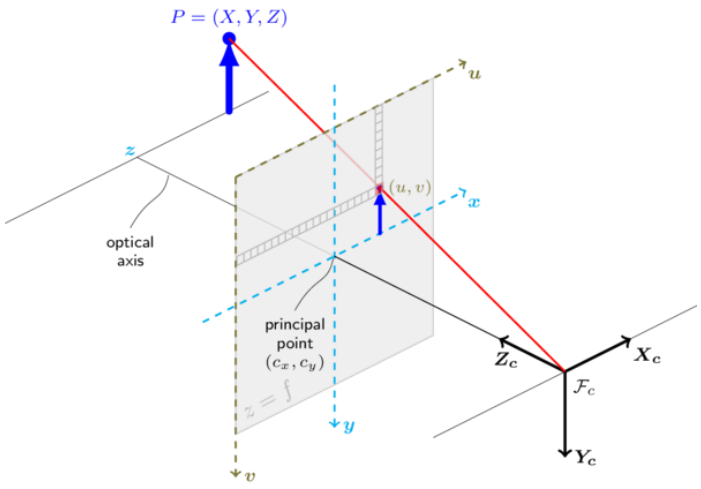


Figure 8 Pinhole camera model

$$K = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix}$$

$fx/z = (u-cx)/x$
$fy/z = (v-cy)/y$

so:

$x = (u-cx)*z/fx$
$y = (u-cy)*z/fy$

3.1.4 Camera Coordinate to World Coordinate conversion:

This is done by finding the camera pose which transforms the word origin to camera origin. In our case the camera coordinate of 1st frame is taken as world coordinate. The camera pose is represented as rigid transformation in 3D given by matrix:

$$[x\ y\ z] = R*[X\ Y\ Z] + t$$

This matrix is a combination of 3d translation and rotation as:

$CM = R*T$ *where R is rotation matrix and T is translation.*

So, world coordinates are computed from camera coordinates as:
$[X, Y, Z]^T = CM^{-1}[[x, y, z]^T$

This whole procedure can be summarized as:
$[X, Y, Z]^T = CM^{-1}K^{-1}$ *[u, v, depth to meters(kinect_depth)]*^{T}

3.2 Mathematical model:

We have rotated the object about its central plane parallel to Y-axis to capture the different depth frames. This procedure seems to have no change in the camera pose, But the rotation of the object in one direction is equivalent to Kinect being rotated in the opposite direction due to their relative motion.

Also, we have chosen the 1st frame camera coordinate as the World coordinate, and the camera is subject to rotation about the plane parallel to its Y-axis, hence the y-axis never changes. This again, removes translation from the equation.

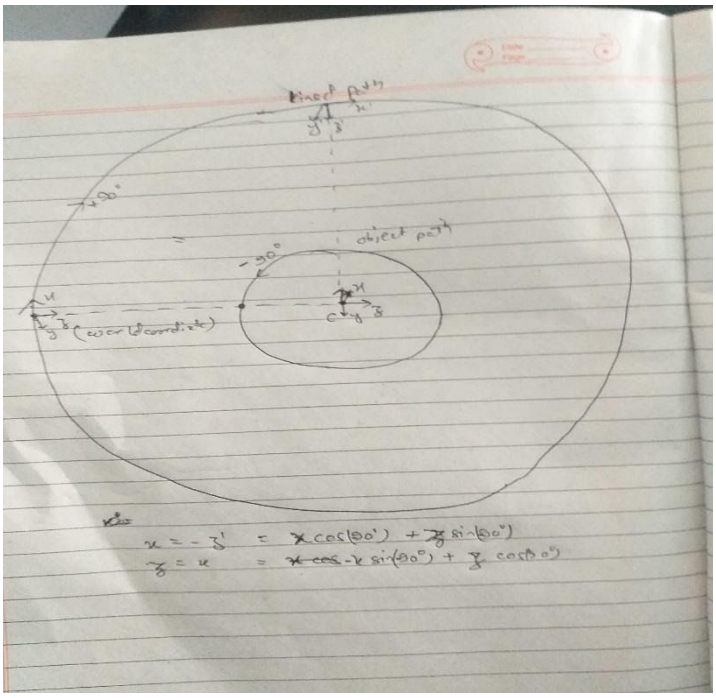


Figure 9 Mathematical Model Development

We can find that the anticlockwise rotation of object by an angle theta(-ve) corresponds to the clockwise rotation of Kinect by theta (+ve) and needs to be compensated by anticlockwise rotation i.e, -ve theta. This is accomplished by

- First translating the center of the object to the origin
  x= x
  y= y
  z= z – c

- Then, rotating the points by theta:
  x= xcos(-thetha)+zsin(-thetha) = xcos(thetha)-xsin(thetha)
  y= y
  z= -xsin(-thetha)+zcos(-thetha) = xsin(thetha)+zcos(thetha)

- Now, translating the points back:

  $x = x$

  $y = y$

  $z = z + c$

4 Epilogue

4.1 Results:

The merged 3d point cloud of the object of interest were generated with persistent misalignments.

Figure 10 Object of Interest

Figure 11 Output (center 0.6m)

After calibration of the centre distance (0.6m to 0.54m), the output was refined.

Figure 12 Output (center 0.54m)

4.2 Discussion

This report thus solves the problem of simplifying the 3d reconstruction pipeline as the proposed system renders a reconstructed point clout of the object of interest. This simplification comes at the cost of perfect alignment of the point clouds from

different views. The reconstruction process actually bypasses various standard steps making this process fast but inefficient.

The translational misalignment is a result of the difference in origin of depth camera axis and the base of Kinect. It can be removed by estimating the camera pose from Kinect base coordinates. Though theoretically sound, it is hardly possible to replicate the perfect version of this experiment resulting in perfect output. The resulting misalignment errors are majorly due to data errors in the mechanical system.

The holes in the reconstruction can be removed using proper filtering at the data refinement step. The outliers in the output needs to be trimmed out properly.

4.3 Conclusion

For this project, we examined the existing 3d Reconstruction pipelines and reduced it down to the proposed model. Then we used our model to create a complete 3d reconstruction of a simple object.

The 3d reconstruction pipeline is hence simplified and made easy by this report at the expense of quality in output and efficiency.

4.4 Recommendation

This system is not recommended for research and practical purposes because of the errors in output. This system can be very useful for learning, experimenting and understanding purposes.

For anyone trying to use this system, they should at first construct a robust mechanical system for taking data to restrict the camera pose as desired. They are recommended to use filters like bilateral filter at the data refinement step to improve the quality of input. They should also use trimming filters for the outliers in the output. Also, a proper care should be taken to find the exact center of the platform

from the pinhole of depth camera as any error in this will cause misalignment and translational error in the output. They are suggested to transform the depth-camera axis to the base of Kinect beforehand to remove the center calibration problem.

5 BIBLIOGRAPHY

[1] Beňo, Peter & Duchoň, František & Tölgyessy, Michal & Hubinský, Peter & Kajan, Martin. (2014). 3D map reconstruction with sensor Kinect Searching for solution applicable to small mobile robots. 23rd International Conference on Robotics in Alpe-Adria-Danube Region, IEEE RAAD 2014 - Conference Proceedings. 10.1109/RAAD.2014.7002252.

[2] van Riel, Sjoerd. (2016). Exploring the use of 3D GIS as an analytical tool in archaeological excavation practice. 10.13140/RG.2.1.4738.2643.

[3] Thoeni, Klaus & Giacomini, Anna. (2012). Efficient Photogrammetric Reconstruction of Highwalls in Open Pit Coal Mines. 85–90.

[4] Klemens Jharmann et. al. (2013). 3D Reconstruction with the Kinect-Camera

[5] Srikanth Varanasi, Vinay Kanth Devu et. al. (2016). 3D Object Reconstruction Using XBOX Kinect v2.0

[6] Haojie Duan (2019). Research on 3D reconstruction technology based on Kinect

[7] Chia-Chi Hsu,Huang Chung-Lin. (2016). Handheld 3D Scanner for Small Objects Using Kinect. Journal of electronic science and technology, 14(4): 377-383.

[8] Özbay, Erdal & Çinar, Ahmet. (2013). 3D Reconstruction Technique with Kinect and Point Cloud Computing.

[9] En.wikipedia.org. (2019). 3D reconstruction. [online] Available at: https://en.wikipedia.org/wiki/3D_reconstruction [Accessed 29 Dec. 2019].