Real-Time Interactive 3D Reconstruction of Indoor Environments With High Accuracy

Shakil Ahmed

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science (Computer Science) at

Concordia University

Montréal, Québec, Canada

November 2020

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Shakil Ahmed**

Entitled: **Real-Time Interactive 3D Reconstruction of Indoor Environments With High Accuracy**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Andrew Delong

_____Examiner
Dr. Andrew Delong

_____ Examiner
Dr. Sudhir Mudur

_____ Supervisor
Dr. Tiberiu Popa

Approved by _____
Dr. Lata Narayanan, Chair
Department of Computer Science and Software Engineering

November 2020

_____
Dr. Mourad Debbabi, Interim Dean
Gina Cody School of Engineering and Computer Science

Abstract

Real-Time Interactive 3D Reconstruction of Indoor Environments With High Accuracy

Shakil Ahmed

3D registration of depth images has been studied extensively in the past. With emergence of low cost RGB-D cameras, many applications have emerged. Yet, the quality of alignment has much to improve. It remains a challenge to create a 3D model of the environment with high accuracy which could be used for engineering applications. Moreover, challenging scenarios where features are scarce, handling large areas as scene, enabling the user to freely roam around the scene for image acquisition in a practical manner, guiding the user in taking better images and accomplishing all of this with low cost RGB-D camera in real time is a subject which is yet to be improved. In this thesis, we address all this challenges by designing and implementing a mobile system that rely on markers. Our system detects and matches markers in real-time with very low CPU load. Moreover, this mobile 3D reconstruction system is interactive which enables a user not only to move freely while doing 3D reconstruction, but also makes the user aware of current status of 3D reconstruction in real-time.

Acknowledgments

I would like to thank my supervisor Dr. Tiberiu Popa. I am thankful to him for guiding me in every step of my masters. I am grateful for having his support and faith on me. I consider myself very lucky to work under his supervision.

I am grateful to my parents and my siblings. Their encouragement and teachings have made my journey possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A picture is worth a thousand words. From cavemen to modern architects, humans have always been trying to describe objects through images. Describing objects through 3D images is even more important as it enables us to represent objects as they are. Therefore, the applications concerning 3D images are broad. From computer aided design in engineering to computer assisted surgery in health sector, the use of 3D representation is extensive. While some applications only need the range image for visualization, others need detailed 3D model of the object or scene. To complete a 3D image of a scene or an object, multiple images are taken from various angles and then aligned properly. For 3D model an extra step of reconstructing the surface is also required. This process has resulted in a pipeline of 3D acquisition, 3D alignment and surface reconstruction. This thesis focuses on the first two. 3D acquisition and alignment.

The nature of 3D acquisition depends on the application need. Sometimes we need detailed and accurate geometry of an object for applications such as digitization of historic artifacts, industrial design, etc. When we have such constraint on geometric data, 3D scanners are the better choice. These triangulation-based scanners use laser light to measure the depth of a point in the scene. Upon shining a laser beam, the camera looks for the spot where the laser hit. Depending on the distance of the surface, the laser spot hits in unique location in the FOV of the camera. The laser emitter, the camera and the point the laser hits, forms a triangle. The distance between the camera and the laser emitter is known. Also, the angle of the laser beam and the angle of the camera corner is also

known through the location of the laser spot in camera FOV. Therefore, the side formed by a point on the surface and the camera can be calculated. Usually, these scanners are fixed in position and can be place in indoor rooms. These scanners are great for scanning small and medium size objects. But they are not suitable for other application where we need to scan large structures from greater distance due to their design.

We need 3D model of large infrastructures or some part of them such as buildings, factories, bridges, etc. for optimization of modification or maintenance applications. Accurate 3D model of such structures saves engineers a lot of time and make their job efficient. 3D models allow them to work in an interactive and collaborative manner. However, for many existing structures, the 3D model was not created at their construction time. Moreover, updated 3D models are needed to learn the current state of the structure especially critical ones. For these reasons these structures are scanned frequently and afterwards analyzed. Analysing these 3D models requires comprehensive details of the structure site. In such cases Lidar scanners are used to capture the current state of the structure to determine if they need any maintenance and how to proceed such tasks. Lidar sensor emits pulsed light waves into the environment. These light waves bounce off the objects and return. The time is recorded for the travel of the light waves. As, the speed of the light is known, the distance is calculated from the recorded time. Lidar sensors can emit such light waves millions of times per second. As a result, it achieves a detailed and precise geometry of the surrounding environment. To move these Lidar scanners a combination of ground robots or aerial drones can be used to acquire data accurately in a synchronized fashion in real time. Usually, these commercial projects are very expensive as it employs a combination of human, Lidar sensors and robots.

While researchers and enterprises using 3D scanners for a while, the demand for everyday use of 3D scanning by ordinary individuals has increased in the past decade and only looks to increase further. Unlike the uses of 3D scanners mentioned earlier, applications used by daily consumers have different requirement for 3D scanners. These applications involve Augmented reality gaming in indoors, various home improvement apps, recommendation apps for wearable apparels, physical

activity recognition, gesture recognition, commercial uses in health care and in educational institutions. These kinds of applications are often done in real time which requires moving and rotating the scanners with ease. Also, the set up of the scanners must be easy and quick. Moreover, the affordability of the scanners must be kept in mind because these applications are to be used by masses. The above-mentioned scanners (3D scanners and Lidar) can not play this role. Therefore, a new type of scanners has emerged. These scanners are termed as depth camera which is smaller in size and thus can be moved and rotated by hand with ease, easy to set up and affordable to consumers. While depth cameras are not suitable for taking detailed geometry of the scene unlike Lidar or 3D scanners, it is well suited for everyday indoor use. In the past decade, enterprises came up with various products. Such as Kinect from Microsoft, Tango from Google, RealSense from Intel, etc. These products were built for consumers for daily use. Moreover, with time depth cameras are becoming more accurate in taking geometry of the scene. To take advantage of this improvement many small enterprises are trying out depth cameras instead of more expensive 3D scanner to see if that can meet their need. One of the applications involve 3D scanning of indoor of buildings, stations, drainage system, etc. for restoration, maintenance or documentation.

Many times, the place where the scanning takes place, is difficult to access (such as tunnels, manholes, etc.). So, it is very inconvenient to go back and forth and scan the area again and again. Therefore, it is very important that the survey is done accurately and efficiently in the first time. An efficient reconstruction means that we will be able to reconstruct the scene with least number of poses. But we also need enough number of poses from various angles and have enough overlapping between them so that the reconstruction of the structures is accurate. A brute force approach would be taking numerous numbers of pictures and hoping that we will end up with enough data. This kind of approach makes the scanning process lengthy and accumulates enormous amount of data which is hard to process. On the other hand, an accurate reconstruction requires that the poses taken during the scanning contains accurate measurement. While using depth cameras which are cost friendly options, they are not as accurate as other costly options like 3D scanner or Lidar. As we can move around and rotate depth cameras very easily, the accuracy of the measurement also depends on the human user or robot who is moving the camera. Moreover, sometimes it is hard to reconstruct the

poses because there is a lack of texture or geometry in the environment. All these challenges make 3D scanning and registration difficult in indoors places such as rooms with not enough textures in the wall.

In our work we addressed challenges mentioned above. We propose a robust solution to make 3D reconstruction process more efficient with a real time registration feedback system. Our method enables the user to determine which poses are to keep and which are to discard during reconstruction. Our system is deployed on a freely moving mobile platform which makes it easier to reconstruct the scene in real time while not being restricted within a certain area. This platform is smaller in size, lightweight and fully equipped with all the devices necessary to take poses and compute the reconstruction.

1.1 Contribution

In this thesis we made a 3D reconstruction system which is accurate in terms of visual but also in terms measurements accuracy in the scene. We tackled these challenges in the following way:

- We developed a mobile platform to conduct real time 3D reconstruction in environments with less features with the help of markers. This mobile platform consists of all the things a user needs to complete 3D reconstruction onsite. To compare the robustness and accuracy of our 3D reconstruction system, we also have developed an offline marker-less 3D reconstruction system. Comparing the result, we show that our system performs better in terms of accuracy and robustness while being an online system deployed on low configuration computer.

- We experimented on the relationship of accuracy measurements in the scene with respect to camera pose from different angle. We show that camera angle influences the measurements in the scene. Based on our findings, we made our reconstruction system interactive which aids the user to take optimal camera pose to acquire more accurate measurement of the scene.

Chapter 2

Background

2.1 RGB-D Camera

This section introduces new Microsoft azure Kinect that was used in this project. We shortly discuss its various features. Azure Kinect is equipped with one RGB-D camera, one ToF depth camera, one embedded Inertial Measurement Unit (IMU) which includes both accelerometer and a gyroscope and a microphone array. The device is recommended to operate in 10-25°C. For 3d reconstruction we only used RGB-D camera and depth camera. Therefore, we will only discuss about these two features.

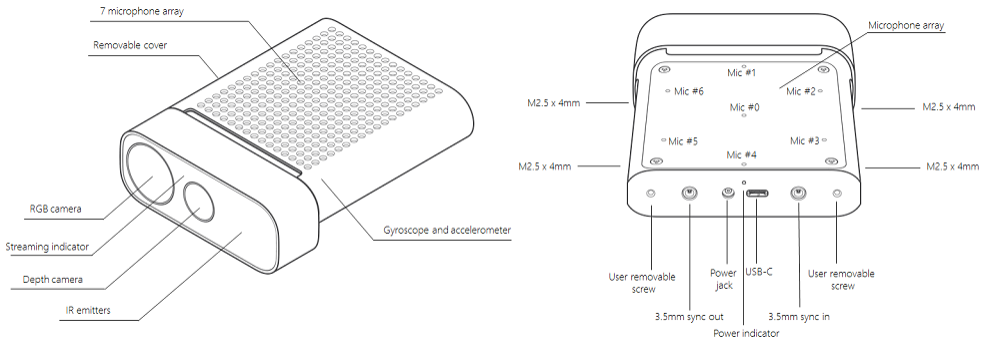


Figure 2.1: Kinect device

2.1.1 RGB Camera

Table 2.1 illustrates different color camera configurations.

Table 2.1: RGB Camera Configurations (Microsoft, 2019).

Resolution	Aspect Ration	Frame Rate
3840×2160	16:9	0, 5, 15, 30
2560×1440	16:9	0, 5, 15, 30
1920×1080	16:9	0, 5, 15, 30
1280×3072	4:3	0, 5, 15
2048×1536	4:3	0, 5, 15, 30

2.1.2 Depth Camera

The depth camera uses time of flight principle for measuring depth. The camera project near infrared spectrum unto the scene. It measures the time for the light to travel to the object and return to the camera. From the time information we can easily get the distance as we already know the velocity of light. From the measurements of distance, it generates a depth map. A depth map contains the distances (z values) for each pixel in millimeters. We also get an IR image which is proportional to the intensity of the light.

It has multiple modes of in terms of the ability to capture image as difference distances. Narrow Field of View (NFOV) mode allows to capture objects from greater distance while having shorter resolution while Wide Field of View (WFOV) are better for capturing images from shorter distances but with wide field of view. The camera also has binning option. The depth camera also comes with binning mode. Pixel binning is a way combining the regular pixels into a super pixel. Such that, the pixels can now absorb more light and thus gives better picture quality. Kinect depth camera offers 2×2 binning which means each pixel in binning mode can absorb four times the light than

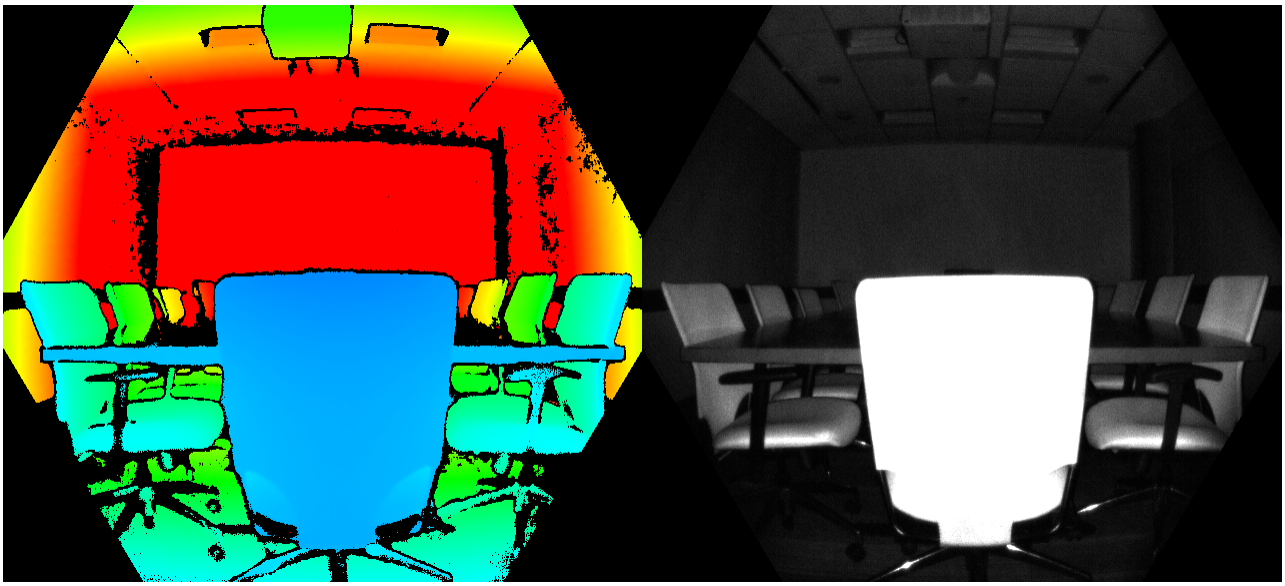


Figure 2.2: Depth Map(left) and IR image (right)

the regular pixels and thus enabling it to measure from higher depth. But the natural drawback is, it reduces the resolution of the image.

Table 2.2 illustrates different depth camera configurations.

Table 2.2: Depth Camera Configurations (Microsoft, 2019).

Mode	Resolution	Frame Rate	Operating Range
NFOV unbinned	640×540	0, 5, 15, 30	0.5 - 3.86 m
NFOV 2×2 binned (SW)	320×288	0, 5, 15, 30	0.5 - 5.46 m
WFOV 2×2 binned	512×512	0, 5, 15, 30	0.25 - 2.88 m
WFOV unbinned	1024×1024	0, 5, 15	0.25 - 2.21 m

Where NFOV means Narrow field-of-view depth mode and WFOV means Wide field-of-view depth mode.

2.1.3 Measurement Error

In various conditions depth camera fails to give correct or valid depth values from some part of the scene. When generating the depth map from IR image in the PC, all invalid depth values are given the value 0.

- If the depth data was capture from an area which is outside of the scope of IR projected area then the depth values, we will get from those part will not be valid.
- If the scene is illuminated with intensive projective light, then there is a possibility that the scene may be saturated with IR light. In such cases, we may get invalid depth. On the other hand, if the scene is very lowly illuminated, IR signal becomes weak to generate depth. It also results in invalid depth.
- A common problem for ToF cameras is multi path detection. Kinect is no exception. Multi path detection occurs when a pixel receives signal from multiple area of the scene. It results in incorrect depth value.

2.2 Markers

Where there are not enough natural features, artificial or fiducial markers can be used to aide the alignment of the poses taken during scanning. These fiducials are reliable and can be detected with high accuracy compared to natural features. Also, the detection rate is real time. Additionally, they can be automatically detected and localized in low resolution even from long range. Moreover, due to the fact that they can generate unique features in different rotations, they are used in tons of computer vision applications such as augmented reality, object recognition, object tracking, camera calibration, SLAM, robot navigation, etc. where robustness and precision is much needed. Their ability to generate unique markers enable alignment of poses taking during scanning robust and accurate. Also, the ability to be detected automatically in real time aide in the real time alignment of poses. Apart from that during experimentation they can act as ground truth and simplify the development process. Any fiducial system must have unique pattern so that it distinguishes itself from any other pattern present in the scene. Such patterns often include repeatable dots, circles or

quadratic shapes. Each marker in the system should be distinguishable from each other too so that they can be identified uniquely. For such a system to be feasible there should be enough number of markers to meet the need of users. There are two main part in a fiducial system. In first part unique patterns for each marker is encoded. Then the patterns are ready to be used in the scene. In the second part, the patterns are identified in the scene while taking poses.

Most popular among fiducials are ARTag (Fiala, 2005), AprilTag (Olson, 2011)(Wang and Olson, 2016), CALTag (Atcheson et al., 2010), etc. These tags have their advantages and weaknesses. There are some criteria from which we can choose which type of fiducials we want. Namely, false positive rate, minimal marker size, marker library size, immunity to lighting condition, immunity to occlusion, speed performance. The false positive rate increases when there exists very similar pattern to the fiducial maker in the scene and also the identification system is not robust. The rest of the criteria contribute to false negative. Marker size plays a role when scanning from a distance. If the marker size is too small then from distance, markers may not be detected. In challenging scenes where objects are dynamic, occlusion of markers may occur. Therefore, ideal fiducial system should be robust against partial occlusion. In general, most of the fiducials work better in ambient lightening. In case of projectile lightening, fiducials tend to give poor result. The above-mentioned fiducial systems give real time identification of the markers. But for the other criteria, each of them differs. (Shabalina et al., 2018) have compared the occlusion aspect of ARTag, AprilTag and CALTag. They demonstrated the size of the markers plays a crucial role in real life settings for identification of marker from distance. In that case, AprilTag seemed to give better result. Also, AprilTag is an open source library publicly available in C++. Which suits our purpose.

AprilTag is a bi-tonal fiducial. That means it only has black and white color. There are two major aspect of designing AprilTag system. Detection or identification stage and encoding stage. The first step of detection step is to detect line segments of quads. Gradient direction and magnitude are calculated for each pixel and then enrolled all pixel in the same connected component with same gradient direction and magnitude. This kind of gradient based clustering algorithm is sensitive to noise therefore, a low pass filter is used. The edges of markers are large-scale features compared to data inside the marker, so it does not result in loss of data. The direction of the line segments is determined by the gradient direction. Which means part of the marker which was white is on

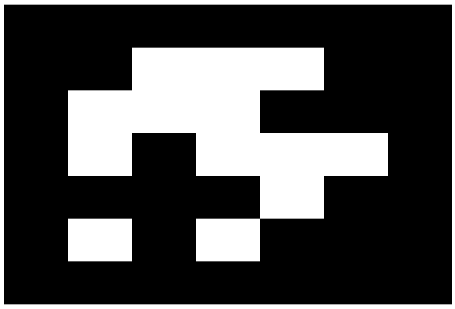


Figure 2.3: An AprilTag Marker

the right side of the line segments and on the left side black parts of the marker. This helps in detecting the quads in the next step. Because, the direction of the segments is already known, a depth first search of depth four, is used to find all three four sides of a quad. In each step of finding a quad, a line segment which already is not part of any other quad is chosen. Then, all the line segment which are close by fulfilling a distance threshold and obey counter clockwise direction are selected as candidates. After completing depth first search, we end up with a candidate quad. Final step of detection is payload decoding. Each quad represents a bit. From the quads we get a vector representing the marker. This vector is compared with all other markers possibly produced by the same family of AprilTag fiducial system. To find the distances between vectors XOR operation is used.

AprilTag employs lexicode system for encoding the markers or tags. In AprilTag a family of markers consisting 36 bits having a minimum hamming distance of 10 is denoted by 36h10. To reduce internal confusion in time of detection, the minimum hamming distance will have to be maintained while markers are in different rotations. Four variation of rotation are considered. 0, 90, 180 and 270 degrees. To reduce false positive cases, a minimum number of 10 quads is considered for any family of AprilTag. This number was selected through experimentation. The assumption is that, complex pattern occurs less frequently in nature. So, the greater number of quads are present in a marker, the more complex the pattern becomes. Encoding markers is computationally expensive, but because it is done offline it is not an issue during scanning.

2.3 Local Feature Extraction

Point clouds descriptors from feature points are used for various computer vision applications. We can categorize 3D point cloud descriptors into two categories. Local descriptors and global descriptors. Global descriptors encode the geometry of the whole point cloud. They are therefore useful for object recognition, object classification, shape retrieval, etc. On the other hand, local descriptors encode geometry of local spaces of the point cloud. Local descriptors can be used for feature matching for registration of the point clouds, object recognition, etc. One of the simple ways of creating these local descriptors is to calculate surface normals or curvature at the feature point. These are local features because normals or curvatures at point are calculated based on their local neighbours. Naturally, these descriptors depend on the size of the neighbourhood and susceptible to noise. On the other hand, depending only one value (ex. normal or curvature) is not discriminating enough to be a proper feature. Therefore, we need local descriptors which are robust to noisy data and outliers, scale and rotation invariant, point cloud density invariant. (Hana et al., 2018) studied various local descriptors in their work. They selected the descriptors based on citation, state of the art performance and their usage in the research and industry. In their experimentation they calculated the accuracy and efficiency of the descriptors based on applying for object recognition. Upon experimentation, they recommend some feature descriptors such as PFH, FPFH, SHOT, CSHOT, etc. For our experiment we have selected FPFH(Rusu et al., 2009) and CSHOT (Tombari et al., 2011).

2.3.1 FPFH

Fast Point Feature Histogram is a local descriptor ideal for 3D registration of point cloud. It tries to describe geometric information of the local neighbourhood of a feature point by generalizing the mean surface curvature around that point. First, surface normals are calculated for every point in the point cloud using principal component analysis. Then for each point p_j in the neighbourhood of p, we follow these steps for every pair (p, p_j):

- Either p or p_j is chosen as source p_s and the other one as target p_t. The one who has the

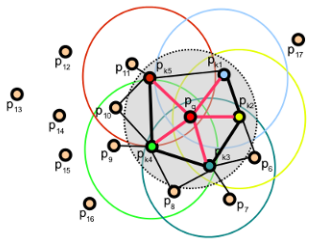


Figure 2.4: Neighbour edges of point p (Rusu, 2019a)

smaller angle between the line segment connecting these two points and its corresponding normal is the source. The other one is the target.

- A Darboux frame is defined at p_s. Let, n_s and n_t are the normals on p_s and p_t respectively. u = n_s, v = (p_t - p_s) $\times$ u and $w = u \times v$

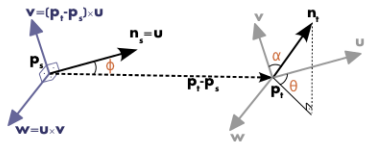


Figure 2.5: PFH Coordinate Frame(Rusu, 2019b)

- Three features are calculated for each pair of p_s and p_t. $f_1 = v \cdot n_t$, $f_2 = u \cdot (p_t - p_s)/(||p_t - p_s||)$ and $f_3 = \arctan(w \cdot n_t, u \cdot n_t)$ .These three features is called Simplified Point Feature (SPF).

- After calculating features (f_1, f_2, f_3) for all of the pairs (p, p_j), three feature ranges are computed (ex: range(f_1) = maximum(f_1) - minimum(f_1)) in the neighbourhood and then divide each of these ranges into five subdivisions. Iterating for each pair, hash each of the features (f_1, f_2, f_3) into one of their own subdivisions or bins. Thus, we have five bins for each feature and in total 15 bins for each point p. This geometric information is represented with three histograms. Together these three histograms for each p is called Simplified Point Feature Histogram (SPFH) of p.

- After calculating SPFH for p, SPFH for the each of the neighbours p_j also calculated w.r.t

their own neighbourhood. SPFH of each p_j is weighted according to their distance from p.

- Finally these weighted SPFH are added to the SPFH of p. Combining both we get Fast Point Feature Histogram (FPFH).

$$FPFH(p) = SPFH(p) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{w_k} \cdot SPFH(p_k)$$

where number of neighbours of p is k and $w_k = ||p - p_j||$.

After getting FPFH for all feature points, the set of points with most discriminating features p_f are selected. We calculate the mean of the feature histograms and take the distance (Kullback-Leibler divergence) between the mean histogram and histograms of each point p. These distances are compared and only those points who have distance more than one standard deviation are selected. This process is repeated for different neighbourhood radius. After this step, we have the points which had unique features for different size of neighbourhood.

2.3.2 CSHOT

CSHOT is a feature descriptor designed for surface matching combining both geometric and texture information. It is an intersection between histogram and signature-based descriptor for 3D point cloud. Signature based descriptors are highly descriptive because of their spatially localized information. Usually, this kind of approach describe a neighbourhood or support by defining an invariant local reference frame and computes geometric properties on each of the points of some subset of the neighbourhood. On the other hand, histogram descriptors try to encode local geometry in a aggregated fashion by quantizing geometric measurements which also requires definition of a local reference frame or reference axis, thus loosing some description of organized local geometry while gaining robustness. SHOT descriptors combines signatures and histogram approach by first defining a scale and rotation invariant local reference frame which is also unique compared to its local neighbourhood. These are the steps to compute the local reference frame for a neighbourhood of points $p_i, i = 1, ..., k$ within a radius of R around the point p.

- We compute the normals for all the points in the point cloud by Total Least Square (TLS)

method. First, we compute the weighted co-variance matrix M for k points in the neighbourhood by making point p as the origin.

$$M = \frac{1}{\sum_{i:d_i \leq R}(R - d_i)} \sum_{i:d_i \leq R} (R - d_i)(p_i - p)(p_i - p)^T$$

where d_i is the distance between p_i and p.

- We apply Eigen Value Decomposition (EVD) on co-variance matrix M. Thus, we obtain three orthogonal eigen vectors. These eigen vectors will denote the three axes of the local reference frame.

- the axes are re-oriented with respect to the direction of most of the vectors of the neighbourhood. This reference frame is unique w.r.t the neighbourhood and rotation invariant.

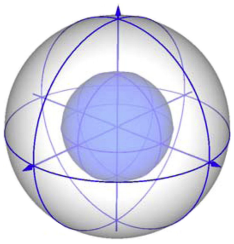


Figure 2.6: Signature Structure (Tombari et al., 2010)

For computing the descriptor of the neighbourhood first local histograms are generated for each of subdivision of the neighbourhood. The subdivisions are created based on the volume and cosine between the normal of the point and reference axis. The volume part is divided by elevation angle, azimuth angle and for the radial distance from the feature point. For getting the cosine for each point in a local histogram, the dot product between the normal on of that point and the surface normal is taken. Using the cosine as feature has two advantage. First is, it is efficient to compute. Second one is because of the nature of cosine function, the points having normals as the same direction as the surface normal are grouped in the same bin. But as the difference increases, the points are binned in different bins. It enables the local histograms to contain fine geometric information

while there is high curvature, while for smooth planar surface region, there is not much diversity in binning. Therefore, the histogram correctly captures the geometric architecture of the local subdivision. Lastly, to avoid the boundary effect, quadrilinear interpolation is used. That is, for each dimension of binning, parameter $1 - d$ is used for interpolation where d is the bin distance for each dimension of bin. Thus, resulting in a weighted histogram. Finally, to avoid the differences due to having point density difference in different poses, histograms are normalized.

2.4 Iterative Closest Point

Iterative Closest Point (ICP) (Besl and McKay, 1992) algorithm is heavily used when it comes to align 3D or 2D poses. Given two point sets $P = p_1, p_2, ..., p_m$ and $Q = q_1, q_2, ..., q_n$ with correspondence $C = i, j$ where $i = 1...n$ and $j = 1...m$, we need to find the 6 DOF transformation in other words rotation R and translation T between P and Q such that:

$$E(R,T) = \sum_{(i,j)\in C} ||q_i - (R \times p_j + T)||^2$$

where $E(R,T)$ is the error function. There are many variant of ICP is available. But the basics steps are as follows:

Algorithm 1: ICP

$error \rightarrow \infty$;

while $error > threshold$ **do**

    Determine corresponding points;

    Compute Rotation R and Translation T;

    Apply R and T to the source point cloud;

    $newError = E(R, T)$;

    if $newError = error$ **then**

        break;

    end

    $error \rightarrow newError$

end

Determining the correspondence points is the most challenging step in ICP. This step is also the most uncertain. Due to the difficulty of finding correspondence many approaches have developed. The first task is to select the points to be used for correspondences. Trivial approach would be to use all the points. But to minimize the error and make the algorithm more efficient we could select a subset of all points. Few criteria are:

- Uniform sub-sampling (Rusinkiewicz and Levoy, 2001) where the picked points are uniformly distributed in the sample and have equal constant weights.
- random sub-sampling (Masuda and Yokoya, 1995) randomly selects multiple sets of inlier points. It takes the best inlier set which give least error at each iteration. It is a very easy implementation of narrowing the subset of points.
- feature-based sampling is efficient and have higher probability of success when the scene contains good geometry or textures.

After selecting the points, we must determine the respective closest points in the target point set. There are many ways we can define what constitutes closeness between two points. Such as distance approach where correspondence is measures on based on the Euclidean distance between the points,

normal shooting approach (Chen and Medioni, 1992) where a normal is shot from source point and the intersected point in the target is chosen as correspondence, close compatible point approach where points have a constraint of having similar color, curvature, features, etc. In our work we have chosen the feature-based approach for sub-sampling of the data and for choosing closeness between points we have chosen compatible point approach as we use feature already. The next step is to determine the transformation matrix between two-point clouds.

2.4.1 Computing Transformation Matrix

To determine the translation, we use the following steps:

- We translate two-point clouds to the origin of a common coordinate system. First, we calculate the centroids μ_P for point set P and μ_Q for point set Q.

$$\mu_P = \frac{1}{|C|} \sum_{(i,j) \in C} P_j; \mu_Q = \frac{1}{|C|} \sum_{(i,j) \in C} Q_i$$

  After that we subtract each centroid from their respective point sets and obtain new translated points sets $P^{'}$ and $Q^{'}$.

$$P^{'} = \{p_j - \mu_P\}; Q^{'} = \{q_i - \mu_Q\}$$

After getting the translation, we use Kabsch algorithm for finding the rotation which minimizes root mean square error between two-point sets. We take the following steps.

- The co-variance matrix of $P^{'}$ and $Q^{'}$ are created.

$$H = \sum_{(i,j) \in C} q_i^{'} \times p_j^{'}$$

- We calculate Singular Value Decomposition (SVD) of co-variance matrix H.

$$H = U \Sigma V^T$$

- We obtain the rotation matrix R as

$$R = V \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{pmatrix} \times U^T$$

  where d = $det(V \times U^T)$

Chapter 3

Related Works

3.1 Measurement Error with Depth Cameras

Measurement error from depth cameras can be systematic or non-systematic. Systematic error is inherent to a camera and depends on its sensors. On the other hand, non-systematic error occurs during image acquisition due to different circumstances of the environment. Where systematic error is specific to a device, non-systematic errors are generalized. Depth cameras measure depth using IR sensors which emits IR signal to the scene. These scanners use either structured light patterns which uses triangulation, or the time signal took to return to the sensor. Either way, the angle signal hits an object matter. In case of triangulation method, (Turk and Levoy, 1994) reasoned that if the signal hits the objects in the scene in grazing angle then, the sensors see the structured light pattern stretched out. This causes discrepancies in depth measurement which adds uncertainty on the accuracy of the depth measurement depending on the viewing angle. (Naik et al., 2011) investigated different environmental effects on depth measurement using ToF cameras. They experimented reflection and refraction properties of emitted IR light hitting objects in different angels. On the other hand, (Khoshelham, 2011) investigated depth error of Kinect due to the distance of the sensor from the scene and the orientation of the surface toward the sensor. In their experiment, they noted that while object surface is oriented at grazing or large incidence angle, a maximum difference of 3cm occurred for 80% of the points pair while data from Kinect was compared with a laser scanner. (Des Bouvrie, 2011) in their work experimented Kinect 2 depth error based on distance from the

objects, viewing angle with respect to the objects, objects with mirroring properties, etc. They measured the average error between the actual measurement and measurement from Kinect 2. In their work, a large discrepancy in depth measurement was found while taking image from greater distance. They also experimented depth measurement based on viewing angle. While there was some discrepancy in that case but compared to distance experiment it was lower. Although, their experiment shows depth error on large incidence angle, they only computed average error, but did not mention the fluctuation of the depth measurements taken in different frames while images taken in different view angels. (Dal Mutto et al., 2012) experimented on depth measurement of Kinect and usability of Kinect data for different computer vision tasks. In their experiments, they found the poor performance of Kinect devices with tilted surfaces. In that case, many depth data were missing in tilted surfaces.

3.2 3D Reconstruction of Scene

In the past decade 3D reconstruction of the indoor scene and objects using RGB-D camera has been extensively studied. Different techniques have so far been used to acquire robust reconstruction. Depending on the scene, primarily two types of reconstruction techniques emerged. Reconstruction with prior knowledge and without prior knowledge.

3.2.1 Reconstruction Based On Prior Knowledge

Sometimes 3D reconstruction is done keeping a scene in mind. For example, 3D reconstruction of a factory setting, merging indoor and outdoor modeling of a site, residential household, etc. Such cases, where much priority is given on the quality of the reconstruction of a scene while losing some generality, using prior knowledge proves to improve the quality of reconstruction. (Son et al., 2015) worked on the highly accurate 3D reconstruction of as-build industrial equipment. In their work they try to model the industrial equipment from 3D point cloud acquired from laser scan. They also use three kind of prior knowledge such as scene knowledge, geometric knowledge and topological knowledge. From scene knowledge they know which equipment are present in the scene. After having the equipment list, 3D dimension of these are acquired as geometric knowledge from

database. The topological knowledge describes adjacency relationship among objects. The 3D reconstruction of instrumentation starts with the segmentation of parts that represent the intersections of the pipelines. Each segment is analysed with respect to their shape and different kind of features are used for different segments. For example, curvature is used as features for pipelines as they are cylindrical in shape.

While some methods depended on the materials presented in the scene, other ones make use of prior knowledge such as assuming the scene was taken in a man-made world which conforms to Manhattan World. As presented in the work of (Coughlan and Yuille, 1999), Manhattan World assumption states that all surfaces in the world are aligned with three dominant direction for example X, Y and Z axes. Using this assumption many later works have tried to estimate the positions of surface walls of the room. The first assumption is that, the scene resides in a Manhattan World and thus the room is cuboid, in other word it has four bounding walls. which means every side of the room can be represented by four corners, again assuming these corners originates from four vanishing points. There have been many works done such as Rother and Carlsson (2001) to find the pose of the camera given the camera parameters and four vanishing points residing in a reference plane visible from all views. This work was further advanced by (Liu et al., 2015) where they proposed a method which enables a 3D virtual tour of apartments. After calculating camera pose through the above-mentioned techniques and assumption, they use the floorplan of the apartments. To reconstruct a whole apartment after reconstructing individual rooms, they rely on floor plans to extract information about doors, windows, etc. to stitch together the rooms into whole apartment. To calculate the relative pose of the room with respect to the apartment, they formulate this problem as inference in Markov Random Field to predict the layout of each room and its relative pose within the full apartment. Later on this work extended by (Wang et al., 2015) to find the camera pose given a particular 2D image of some part of shopping mall. Along with other prior knowledge mentioned before, they utilized texts mainly the name of the shops rather than shape present in the image.

While these model-based reconstruction techniques utilize geometric information and prior knowledge to calculate camera pose with respect to the scene, SLAM or Structure from Motion techniques use multi-view relationship to calculate camera pose and as a result reconstruct 3D map. However, SLAM or Structure from Motion often suffer from drift error accumulated in each step of

calculating camera pose and 3D map simultaneously. To address this problem, many solutions were proposed by utilizing prior knowledge of the scenes. (Tamaazousti et al., 2011) proposed a solution to calculate camera pose in a partially known environment. They combined geometric information from the known model of the scene with the SLAM process. The main idea is to use a combination of two error functions in the local bundle adjustment process. One error function was re-projection error function. Along with minimizing re-projection error, they added another constrained or error function based on the geometry of the known environment. This added constrained is actually a planar constraint which exploits the fact that a 3D point on a plane has only two degree of freedom and therefore this point can move only on its associated plane. Later on, (Melbouci et al., 2016) incorporated depth data in the bundle adjustment process to improve this method using RGB-D cameras.

3.2.2 Reconstruction Without Prior Knowledge

As prior knowledge is often not available when conducting reconstruction of scenes, it is done without any prior knowledge. A common approach was attempted through multi-view stereo technique. This technique has several variations. In terms of photo consistency measure, (De Bonet and Viola, 1999; Seitz and Dyer, 1999) tried to reduce the projection error through the variance of the projected pixels from the scene. Where others have tried window approach of minimizing the squared sum error of particular size of window (Faugeras and Keriven, 1997; Jin et al., 2003). Another problem with stereo is occlusion which happens due to not knowing if a point is visible in a pose. To avoid that ordering or uniqueness constraints were used by (Bobick and Intille, 1999; Ishikawa and Geiger, 1998), implemented by dynamic programming. This method could not solve occlusion when scene has narrow holes or foreground objects are thin. Later, (Sun et al., 2005) improved this problem by symmetric stereo matching modeling using visibility constraint instead. But such techniques lacks robustness and accuracy(Kim and Woo, 2005). Specially, the depth error is larger compared to that of the laser scanner or ToF cameras.

Use of laser scanner for 3D reconstruction is also not new. Most of the experiments regarding laser scanner were done in lab environment before 2000. One of the notable works was done by

(Beraldin et al., 1997) of National Research Council of Canada, where they developed a prototype laser range camera specifically to scan medium to large object and successfully demonstrated the high accuracy of measurement by laser cameras. In 2000, famous digital Michelangelo Project (Levoy et al., 2000) used a combination of both triangulation and time of flight-based laser scanner, as well as digital still camera to acquire images of various statues in a non laboratory environment. After 2010, the evolution of time of flight camera permitted to scan small to medium size objects with a single handheld RGB-D camera despite issues with noise and inaccuracy in depth measurement. Notably, in 2011 (Newcombe et al., 2011) used a single Kinect to scan indoor objects and 3D reconstruct them simultaneously. They developed KinectFusion, a framework for online 3D reconstruction of the indoor scene using depth data for registration rather than RGB data. Their GPU based implementation made the processing very fast. Instead of using feature points, authors used all the 3D points in order to register two scans of inputs. The frame rate of capturing is very high. Therefore, they were able to use a point-plane metric based ICP on all the available 3D points captured in current frame to register current scan with globally aligned scene. The surface of the globally aligned scene is stored implicitly through weighted Truncated Signed Distance Function (TSDF) which was ray casted to predict the surface. The weights, TSDF values corresponding each 3D point sample was stored in a 3D voxel grid. This work can compute a up to dated surface registration on real time. While KinectFusion works well up-to midsize rooms, it faces issues with lack of GPU memory while trying to scan large size rooms. Another problem with KinectFusion is that it cannot recover from a misaligned registration. Furthermore, it does not provide solution for drift errors which occurs while scanning large size scenes. Later on, (Whelan et al., 2012) in their work named Kintinuous, addressed the problem of large-scale reconstruction by using memory consuming TSDF structure only for newer scanning region while keeping rest of the globally aligned scene as triangular mesh also known as moving volume method. The issue with GPU memory was further improved by (Nießner et al., 2013) by introducing voxel hashing which managed to avoid empty spaces in voxel grid containing TSDF values. Some work also has been done in trying to address the loop closure problem with fast registration of the poses. But these works highly depends on the distinctive nature of the features and a certain threshold of overlapping of consecutive poses to maintain

moving volume technique(Whelan et al., 2013). As finding distinctive features is difficult if a system only relies on depth data, others have combined color data with depth data and created a hybrid system. (Henry et al., 2012) used both depth and color data for 3D reconstruction. They improved the accuracy of bundle adjustment by computing re-projection error for frame-to-frame alignment RANSAC which performed better than Euclidean-space RANSAC. They also incorporated scene information through object recognition from bag of visual words techniques which results in better loop closure. To make 3D reconstruction process less computation heavy, they use sparse bundle adjustment. Through such steps, they achieved almost real time reconstruction performance. Although they did not take account cases where depth information was not present. Later on, other works improved this specific case by ignoring depth data and using color data instead (Wang et al., 2014). While this work handled repetitive pattern and partially the depth data but failed to make the system online and also did not address how to handle noisy depth data. However, (Dai et al., 2017) managed to address the issue of drift error with reconstructing larger size room with global pose optimisation working parallel with online reconstruction while using same techniques for surface representation, volume fusion, pose estimation as (Nießner et al., 2013). Traditional SLAM systems consist of optimization, odometry estimation, raycasting and map fusion modules. So far traditional techniques to compute these modules have been non-differentiable. A differential system consists of equations which are differentiable or in other words all of them have partial derivatives at any point. A system must be differentiable in order to apply backpropagation technique which is a crucial step in supervised machine learning algorithms. Recently, (Jatavallabhula et al., 2019) proposed fully differentiable alternative SLAM system that operates on three kind of traditional fusion methods such as voxels, surfels and pointclouds. This differential SLAM system opens the door for gradient based learning for SLAM.

3.2.3 Reconstruction With Markers

Markers have been used for scene reconstruction in various manners. Early works involve using markers in conjunction with natural features present in the scene to make reconstruction more robust. (Klopschitz and Schmalstieg, 2007) attempted a large-scale indoor scene reconstruction using both natural features and fiducials markers from structure from motion point of view. They placed

markers in the scene and recorded a video of the scene. From each frame features are tracked to select views for reconstruction. Amongst those selected frames makers are detected and triangulated to get 3d position. (DeGol et al., 2018) improved this work by implementing incremental structure from motion in large scale. They chose indoor scenes with variety of texture and in some cases lacks in texture. They first developed a system without using fiducial markers and show their system is as good as concurrent state of the art systems. After that they use fiducial markers to improve 3d reconstruction. They show that using markers help with reconstructing scenes with less natural features present. It helps with ordering of frames for structure from motion. Also, the performance of bundle adjustment is enhanced when using marker position for optimizing the camera poses. Markers are also used in various robotics applications. In SLAM problems, one of the critical steps for localization is data association between landmarks. Getting the correct correspondence between two frames becomes challenging if landmarks do not have distinctive features or if the environment is dynamic. To address such challenges, (Lim and Lee, 2009) used fiducial markers as landmarks in their EKF based SLAM. They show the estimated value for the movement of the robot was close to the predicted movement. Furthermore, (Neunert et al., 2016) expanded on that idea by adding IMU sensor for dynamic SLAM system which also leverages fiducial makers AprilTag for not only to compute 3D position of the landmark but also the pose estimation. Although adding IMU sensor created more uncertainty in the system, using AprilTag made their system more robust than other contemporary methods. Later, (Munoz-Salinas et al., 2019) expanded on this work by handling pose ambiguity calculating camera orientation using fiducial markers. They also added bundle adjustment process using fiducial markers positions. For solely 3D reconstruction of indoor scenes, recently (Madeira et al., 2020) used fiducial markers for pose estimation. They used RGB-D camera Google Tango for capturing scenes. Although they used RGB-D data for creating the scene model, they only used color frame for pose estimation as they deemed depth data very noisy and unreliable. Also, their system works in offline, framing the pose estimation as optimization problem. They use camera pose estimation acquired from Google Tango framework as initial guess for relative camera orientation parameters for successive frames and use 3D position of the markers as data point for optimization.

Chapter 4

System Methodology

In this section we will discuss about our proposed methodology. We developed a marker based real time 3D registration and reconstruction system. To make this system mobile and easy to manage, we build a hardware platform setup fitted in a small box so that user can easily move around the scene, acquire pose and align them online. We also developed a feature based offline registration system to compare the results with our marker-based system. First, we discuss about camera calibration and then dive into both systems.

4.1 Camera Calibration and Mapping from Depth to Color Camera

Using provided API by Kinect, we extract the intrinsic parameters of color camera and Depth camera. We also extract camera reference frame transformation matrix T_{dc} from depth to color. We create the point cloud from depth data by un-projecting pixel from depth image using intrinsic parameters of depth camera 4.1. Although we have used Kinect API for that purpose, same thing can be achieved using OpenCV un-project function given camera intrinsic which is open source. Before, taking any images, we precomputed a lookup table by storing x and y scale factors for every pixel of depth map by un-projecting the depth pixels assuming the depth as 1.0 and saved the table. While taking images, we just multiply these scale factors with depth to get the 3d coordinate in depth camera to finally get point cloud. This well-known technique makes generating point cloud very efficient.

Table 4.1: Depth Camera parameters

Parameter Names	Value
Principle Point on X-axis(C_x)	315.45
Principle Point on Y-axis(C_y)	327.292
Focal Length on X-axis(f_x)	505.198
Focal Length on Y-axis(f_y)	505.12
radial distortion coefficient(k_1)	4.16775
radial distortion coefficient(k_2)	2.51986
radial distortion coefficient(k_3)	0.123716
radial distortion coefficient(k_4)	4.50163
radial distortion coefficient(k_5)	3.88686
radial distortion coefficient(k_6)	0.668598
tangential distortion coefficient(p_1)	-8.32685e-05
tangential distortion coefficient(p_2)	5.65498e-05

We use these parameters to map from a 3D point $P_d = (X_d, Y_d, Z_d)$ of depth camera to pixel coordinate (r, c) of color camera. We first transform P_d to the reference frame of color camera $P_c = (X_c, Y_c, Z_c)$.

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = T_{dc} \times \begin{pmatrix} X_d \\ Y_d \\ Z_d \end{pmatrix}$$

$$X_c^{'} = \frac{X_c}{Z_c}$$

$$Y_c^{'} = \frac{Y_c}{Z_c}$$

To tackle radial distortion and tangential distortion we undergo additional calculation.

$$X_c^{''} = X_c^{'} \times \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 X_c^{'} Y_c^{'} + p_2(r^2 + 2X_c^{'2})$$

$$Y_c^{''} = Y_c^{'} \times \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + p_1(r^2 + 2Y_c^{'2}) + 2p_2 X_c^{'} Y_c^{'}$$

$$r^2 = X_c^{'2} + Y_c{'}2$$

where $k_1, k_2, k_3, k_4, k_5, k_6$ are radial distortion coefficient, p_1, p_2 are tangential distortion coefficient. Finally, we project $X_c^{''}$ and $Y_c^{''}$ to image plane in pixel coordinates.

$$r = f_x \times X_c^{''} + c_x$$

$$c = f_y \times Y_c^{''} + c_y$$

where f_x and f_y are focal lengths, c_x and c_y are principle point.

After calibration we have got the following values for the intrinsic parameters of the camera and transformation matrix T_{dc}. The units are in millimeter.

Table 4.2: RGB Camera parameters

Parameter Names	Value
Principle Point on X-axis(C_x)	638.063
Principle Point on Y-axis(C_y)	370.472
Focal Length on X-axis(f_x)	605.601
Focal Length on Y-axis(f_y)	605.547
radial distortion coefficient(k_1)	0.91605
radial distortion coefficient(k_2)	-3.06841
radial distortion coefficient(k_3)	1.67757
radial distortion coefficient(k_4)	0.793718
radial distortion coefficient(k_5)	-2.90741
radial distortion coefficient(k_6)	1.6145
tangential distortion coefficient(p_1)	0.0004797
tangential distortion coefficient(p_2)	0.000103107

$$T_{dc} = \begin{pmatrix} R \\ T \end{pmatrix} = \begin{pmatrix} 0.99999 & -0.00434887 & 0.00109698 \\ 0.00442275 & 0.99678 & 0.0800661 \\ -0.000745252 & -0.0800701 & 0.996789 \\ -32.0152 & -2.47191 & 3.68335 \end{pmatrix}$$

The intrinsic parameters of color camera are given in Table 4.2

As azure Kinect SDK is currently only fully available in windows, having these intrinsic and extrinsic parameters, we are enabled to make our framework platform work for any operating system. Also, given intrinsic parameters and extrinsic parameters of color camera and depth camera, our framework can work with any RGB-D camera available in the market.

4.2 Kinect 3D Measurement Error

The systematic error due to sensor is low in case of Azure Kinect which is less than 11 mm + 0.1% of the distance from object without multi-path interference. Also, Azure Kinect comes with multiple depth image resolution and each depth resolution has its own recommended distance range from objects (Microsoft, 2019). We followed the recommended distance range while image acquisition. Apart from systematic error, environmental factors also influence depth measurement in case of RGB-D cameras as we discussed in 3.1 such as distance from objects, light intensity in the environment and view angle of camera with respect to the object. In case of distance, we did not notice meaningful depth error. Also, we conducted experiments in indoor rooms where there is not much fluctuation of light. Therefore, we did not observe any depth error due to difference in light intensity either. However, while conducting experimentation we observed, depending on the rotation of the camera with respect to the scene, the error in depth measurement was different. Therefore, we conducted an experiment of 3D measurement error for rotation of the camera with respect to the scene.

4.2.1 3D Error With Rotation

In this experiment, we explore how rotation of camera with respect to the object to be scanned effect the three-dimensional measurement. For this experiment we use two AprilTag markers. We attach them on the wall vertically aligned and keep some distance in between as seen as in Fig: 4.1. The green line in the middle of the box is the line we are measuring. But first we take the measurement by tape. From the tape measurement the length of the line is 50 cm which is the ground truth.

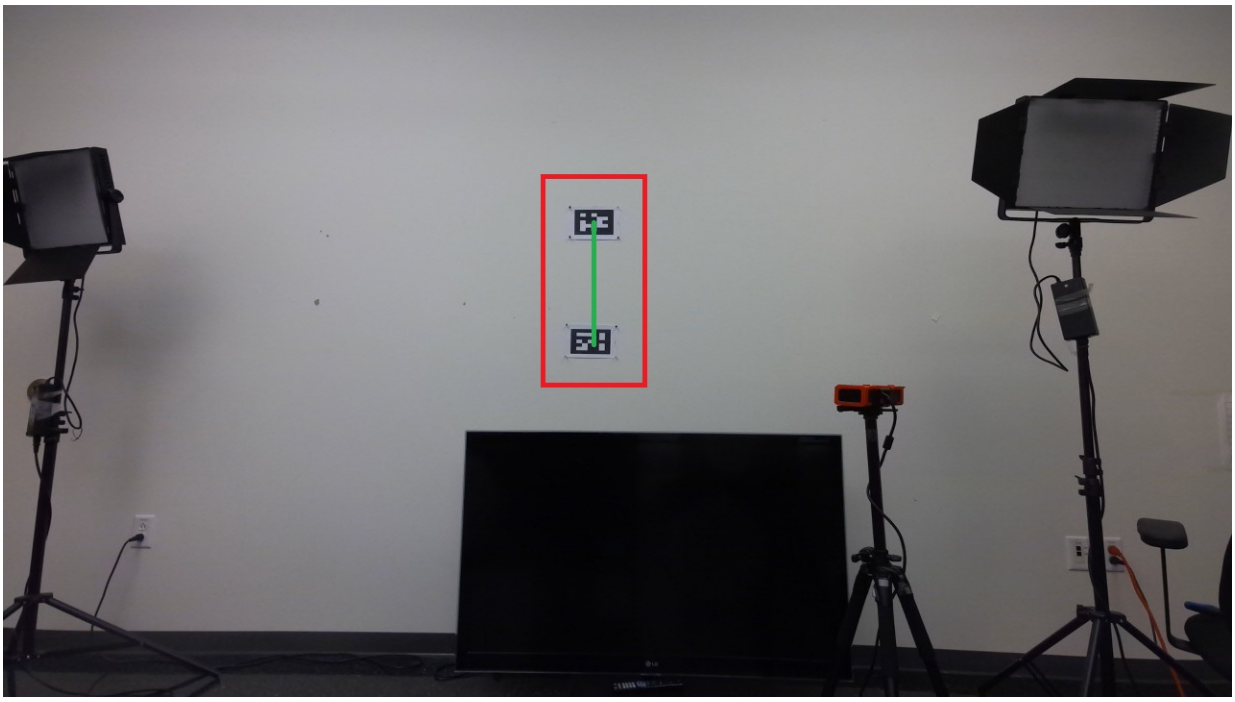


Figure 4.1: 3D error measurement setup

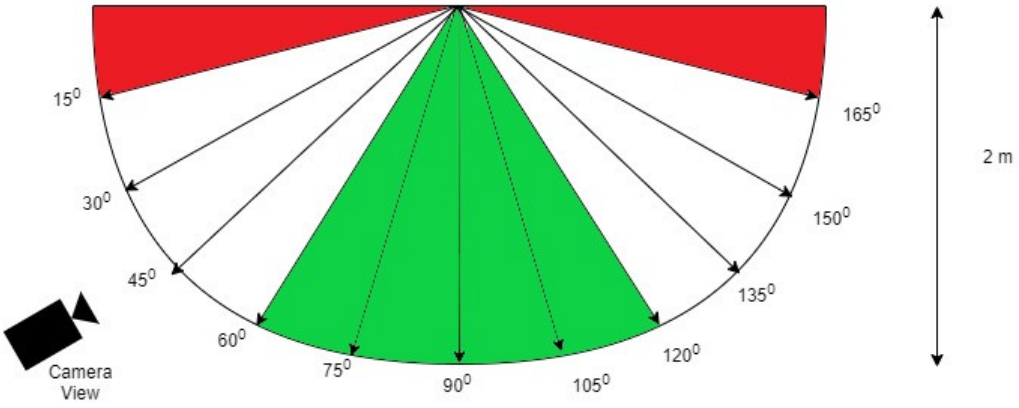


Figure 4.2: 3D error measurement setup (camera angels). Images taken within green angles results in less depth uncertainty while within red area measurement is very uncertain

Afterwards, we take the measurement of the line by taking the difference of the centre points of two markers obtained from Kinect RGB-D images. We rotate the camera with 15°interval from 15°to 165°in anti-clockwise direction at 2 meters as shown in Fig: 4.2. That is, at 90°rotation negative z-axis of camera and normals on the surface of the scene have no angular difference.

At each rotation we take 500 images. From the 500 images of each rotation, we calculate the distribution of the measurements by calculating average and standard deviation. The average of the measurements at each rotation usually converges after 200 to 300 frames.

Table 4.3: Rotation Measurement Error(Original Length = 50 cm)

Angle	Length(cm)	Standard Deviation(cm)
15°	49.55	0.24
30°	50.42	0.21
45°	50.45	0.14
60°	50.31	0.07
75°	50.50	0.07
90°	50.54	0.06
105°	50.55	0.06
120°	50.31	0.08
135°	50.55	0.11
150°	51.05	0.19
165°	50.55	0.25

From table 4.3, we can see measurement differences from various rotation angle. The experiment shows, at both 60°and 120°we got the closest measurement from the ground truth. Also, angles between 60°and 120°shows to have better accuracy in terms of average. The lowest accuracy was observed at 15°and 150°rotation. Although practically there seems not to be much difference in average measurement, in general except for 15°rotation, Kinect always overestimate the measurement.

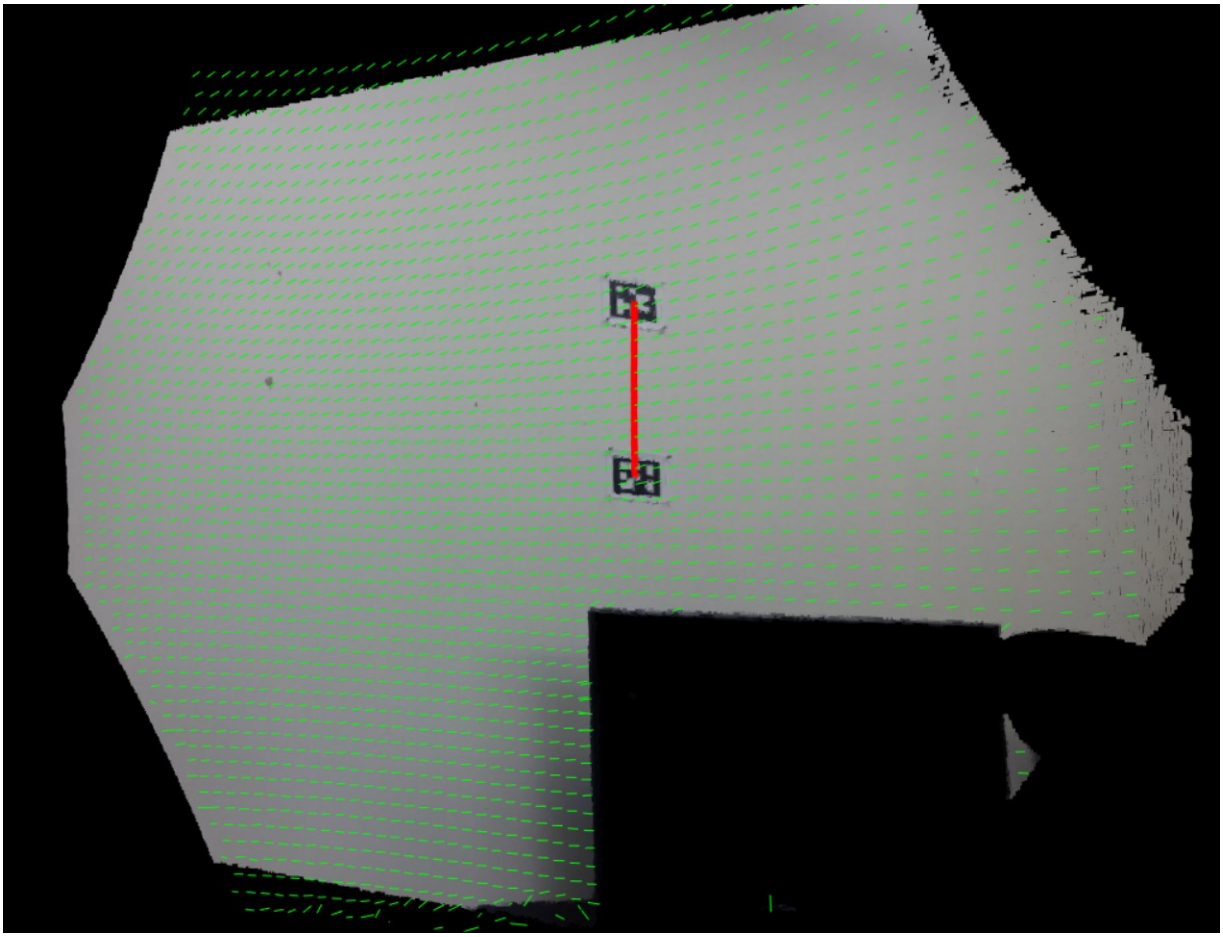


Figure 4.3: 3D error measurement at 30°rotation

In terms of Standard Deviation, it seems to increase as we deviate further from 90°. The highest measurements were observed at 15°rotation and at 165°rotation. Outside of the range of 60°- 120°, the standard deviation doubles. That means the measurement is uncertain outside of 60°- 120°range. Our finding matches with the conventional understanding of ToF cameras. As range data obtained

from ToF cameras have bigger uncertainty when obtained at grazing angles. Therefore, readings from such angles are avoided as suggested by (Turk and Levoy, 1994) and (Naik et al., 2011). From these observations, we can conclude that better measurement consistency can be expected while acquiring images within 60°to 120°range.

4.3 Feature Based 3D Registration

We study the feasibility of 3D scene reconstruction including cases those lack in texture and geometry. First, we conduct feature-based registration which is done offline so that this system is not constrained by time or hardware. In the next sub section, we give an overview of our system.

4.3.1 Overview

The first step of this process is to calibrate the camera and get depth to RGB camera transformation matrix. This part is same as the steps described in 4.1. For the image acquisition, we focused on two scenarios. The first scenario included a scene with plenty of features. That means the scene contains fair amount of texture and geometry so that feature-based algorithm would be able to 3D register the scene with better accuracy. In the second scenario, the scene includes an office space which lacks in texture and geometric diversity. To get a global registration the first stage is to get an accurate pair wise registration between two-point clouds. If the pair wise registration has high accuracy, the global registration becomes easier. Here are the steps we followed to complete the pair wise registration between two-point clouds.

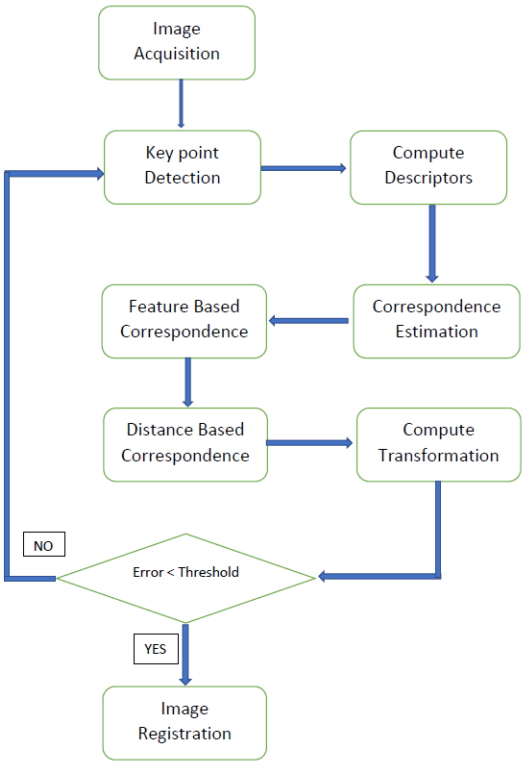


Figure 4.4: Feature based pair wise 3D image registration

4.3.2 Key Point Detection and Feature Descriptors

To use all the points in the point cloud for registration is expensive. Moreover, some points are noise, and some are not accurate. Therefore, not all points are in good position to be used as features. So, determining which points have potentials to be good features is an important step. These useful points are called key points. We used SIFT key point detector (Lowe, 1999) in RGB data and then projected those points in 3D to be used to compute 3D feature descriptors.

(a) (b)

Figure 4.5: SIFT Key-point Detection

We chose Fast Point Feature Histogram (FPFH) (Rusu et al., 2009) and Color Signature of Histograms of OrienTations(CSHOT) (Tombari et al., 2011) as feature descriptors. Both descriptors are scale and rotation invariant. As CSHOT utilize both texture and geometry, we used CSHOT in scenes where both texture and geometry were present. However, we noticed in cases where only geometry was present and texture was lacking, FPFH gave better registration result. The reason CSHOT was giving poor result in texture less scenes are because, CSHOT is depends on texture as much as geometry. Therefore, in texture-less scenes it finds far less feature point correspondences. After feature matching, we got very few point correspondences in case of CSHOT which was not good enough to align two-point clouds as shown in Fig: 4.6.

Figure 4.6: poor performance of CSHOT descriptors in texture-less scene

On the other hand, FPFH only rely on geometric information. Therefore, distance between two feature points in terms of feature description histogram is less in case of FPFH while matching two feature points. From Fig: 4.7 we can see that FPFH succeeds while CSHOT fails in case of scenes with less texture.

Figure 4.7: Successful alignment of point clouds with less texture using FPFH descriptors

4.3.3 Pair Wise and Global Registration

After computing the feature descriptors, we find the correspondence between the features. The correspondence estimation operation is quadratic as we have to compute the match between every possible pair of points between two set of key points. Because, both of these descriptors are represented by histograms, we represent each histogram as vector and compute the correspondence score by a dot product between the histograms. After getting the correspondence score, we select a pair (a, b) if not only a has the best score with b but also, b has the best score with a.

After feature estimation, we remove some of the pairs based on Random Sample Consensus (RANSAC) algorithm. That is, we model the remaining feature point in target point cloud (reference point cloud) and try to detect inliers and outliers in the source point cloud (the cloud which will be transformed) based on that model. After the rejection of outliers, we select the remaining feature points and use to compute transformation matrix using Singular Value Decomposition (SVD). Then

the source cloud is transformed and then we compute the error between feature points the source and target point cloud and take their average. if the average error is below the threshold of 10mm then we stop the iteration, otherwise we start the next iteration. From the next iteration, we add another outlier rejection method based on the distance.

After completing all the pair wise registration, we again employ ICP for global registration. This time we change the error metric to global error.

4.4 Marker Based Real Time 3D Registration

We have developed a real time 3D registration tool with feedback during taking pose for better alignment and accuracy on low cost platform. The flowchart 4.8 shows the steps.

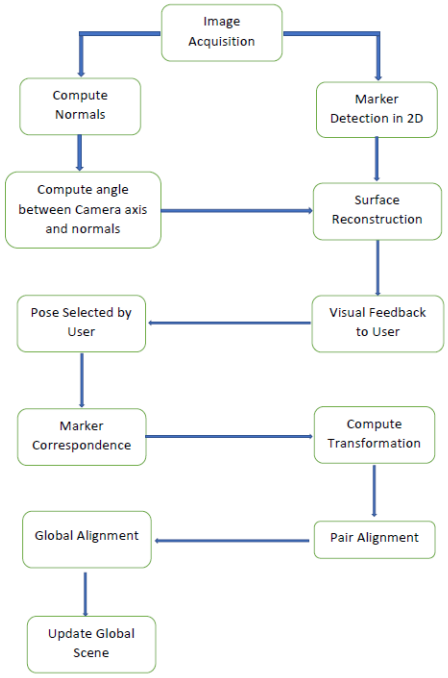


Figure 4.8: Marker Based Registration pipeline

4.4.1 Platform Setup

Our goal is to make this system online and mobile. The online part mostly involves software side where the mobile part is about hardware. To process the computation, we have used a mini PC

stick from Intel with 4 GB RAM, DDR3 atom processor and 64 GB HDD. To power both the mini PC and Kinect we have used 27,000 mAh battery power bank. Also, for displaying the alignment process and let user to interact with GUI, we have used a 6 inch by 4-inch touch display. The platform setup can be viewed from Fig: 4.9, Fig: 4.10 and Fig: 4.11.

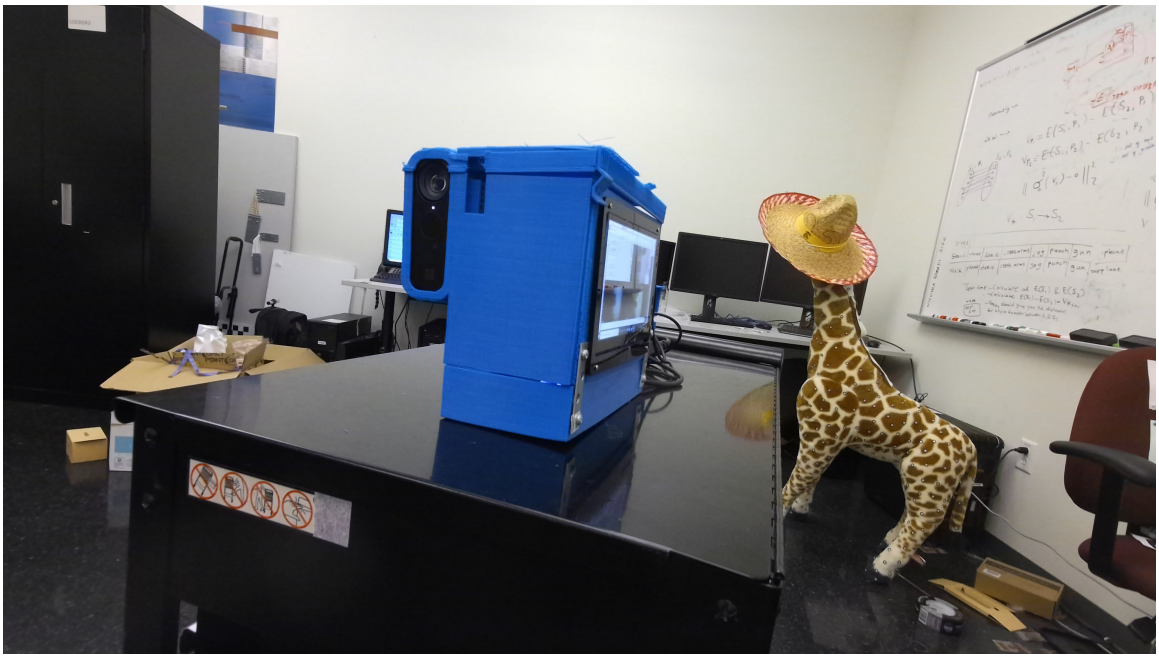


Figure 4.9: Platform Setup Front View

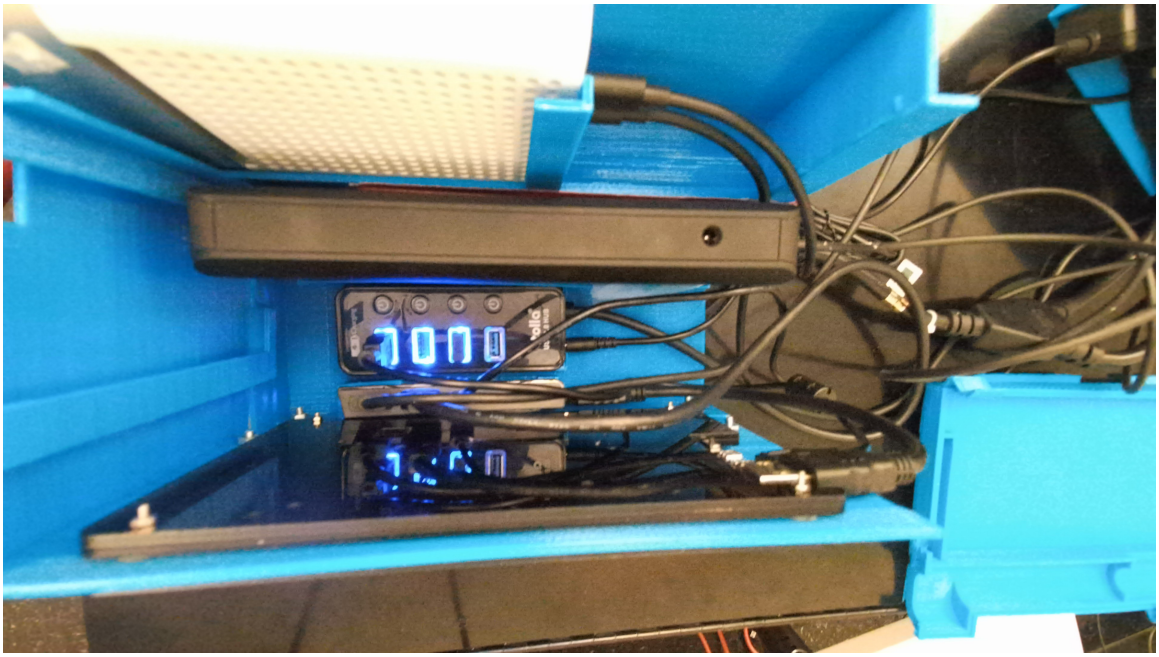


Figure 4.10: Platform Setup Top View

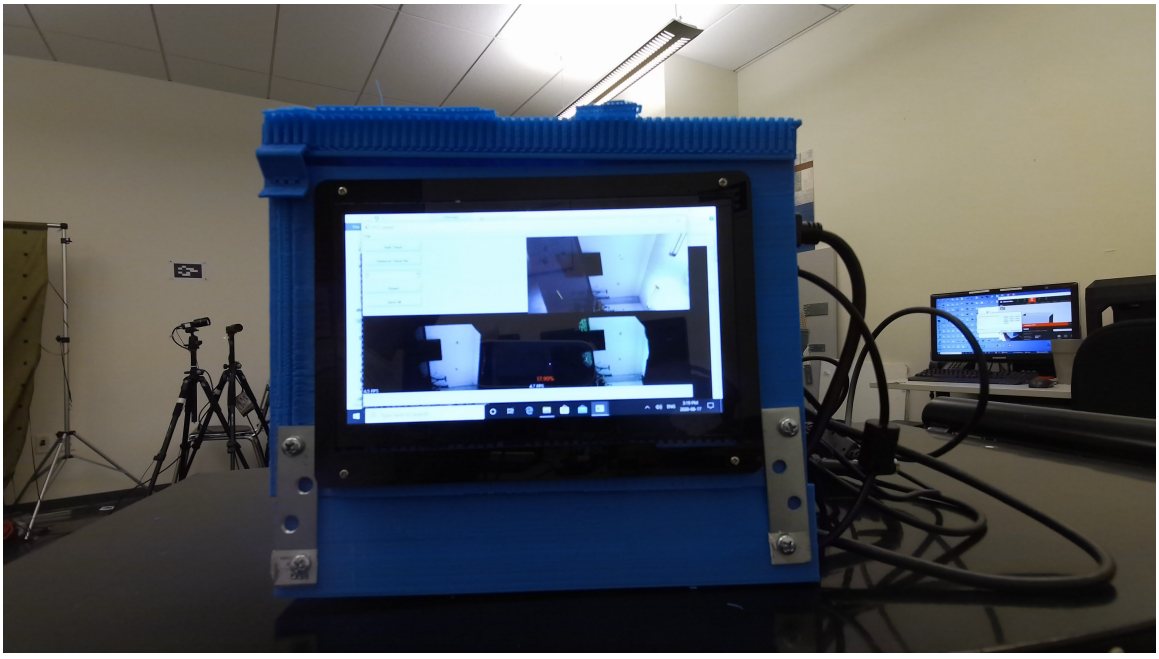


Figure 4.11: Platform Setup Side View

4.4.2 Image Acquisition

Both depth and color cameras are synchronized so that that both images are received at the same time. But the reference frame and resolution of two cameras are different. Therefore, the images need to be aligned. The transformation between depth and color camera frame are known previously as it is provided by the vendor. We created RGB-D point cloud by aligning depth image w.r.t the color image. As Kinect is sensitive to projective light and performs better in environment with ambient light, it was made sure the place has enough lights without sudden changes. There is multiple mode of taking depth image is available in Kinect. We choose NFOV un-binned mode with resolution 640×540 (Table 2.2) as it maintains balance between resolution and depth range.

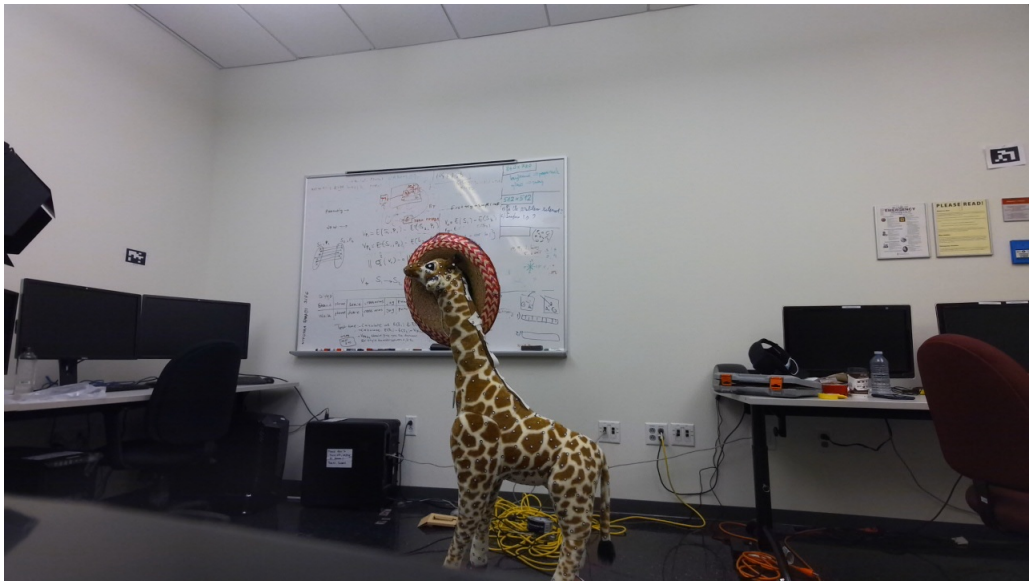


Figure 4.12: RGB image taken from Kinect

4.4.3 Marker Detection and Projection to 3D

AprilTag is a visual fiducial which works in 2D images. From the 2D image we detect the AprilTag markers in the scene. To detect the markers, we have used the library provided by the author of AprilTag (Wang and Olson, 2016). As AprilTag is rotation invariant, we can uniquely identify each corners of a tag. Thus, we select four corners of each tag as our feature points.

But as we are doing the registration with 3D point clouds, we need to get the corresponding 3D

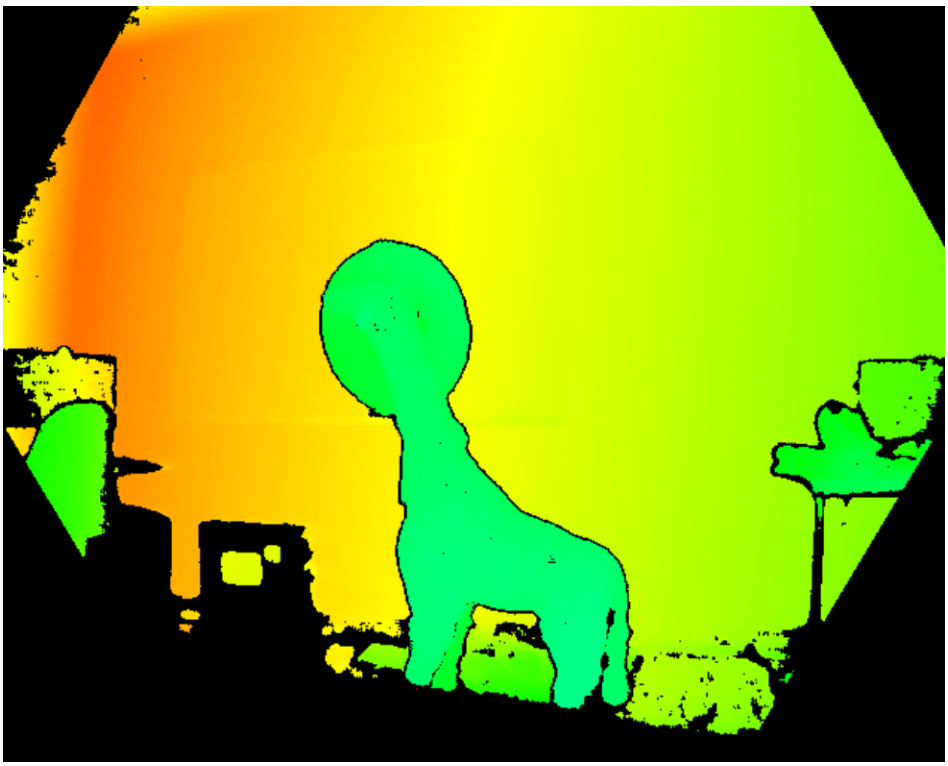


Figure 4.13: Depth image taken from Kinect

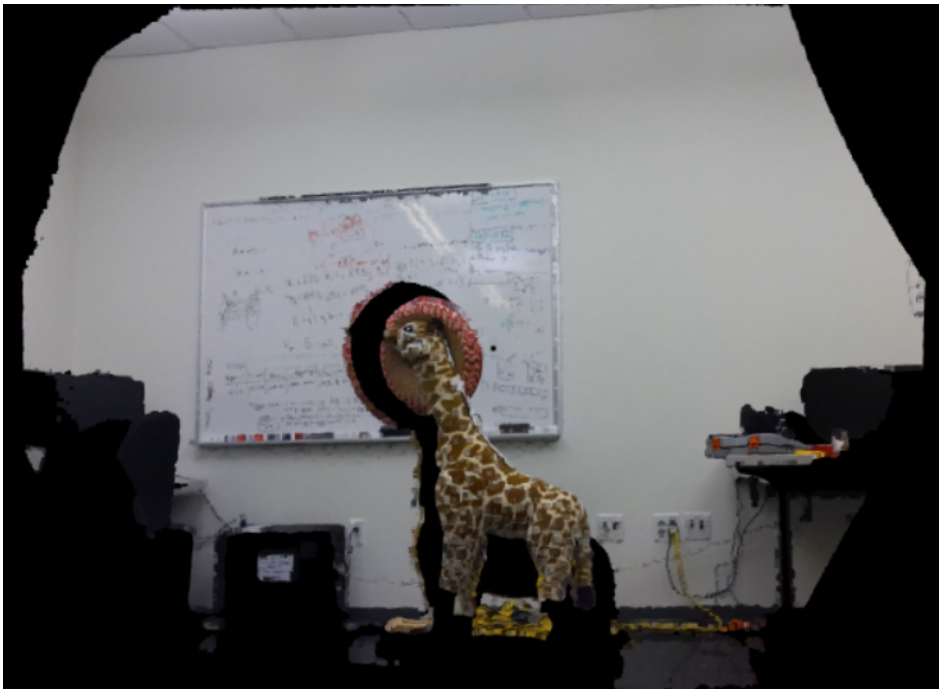


Figure 4.14: Image of Corresponding Point Cloud from 4.1 and 4.2

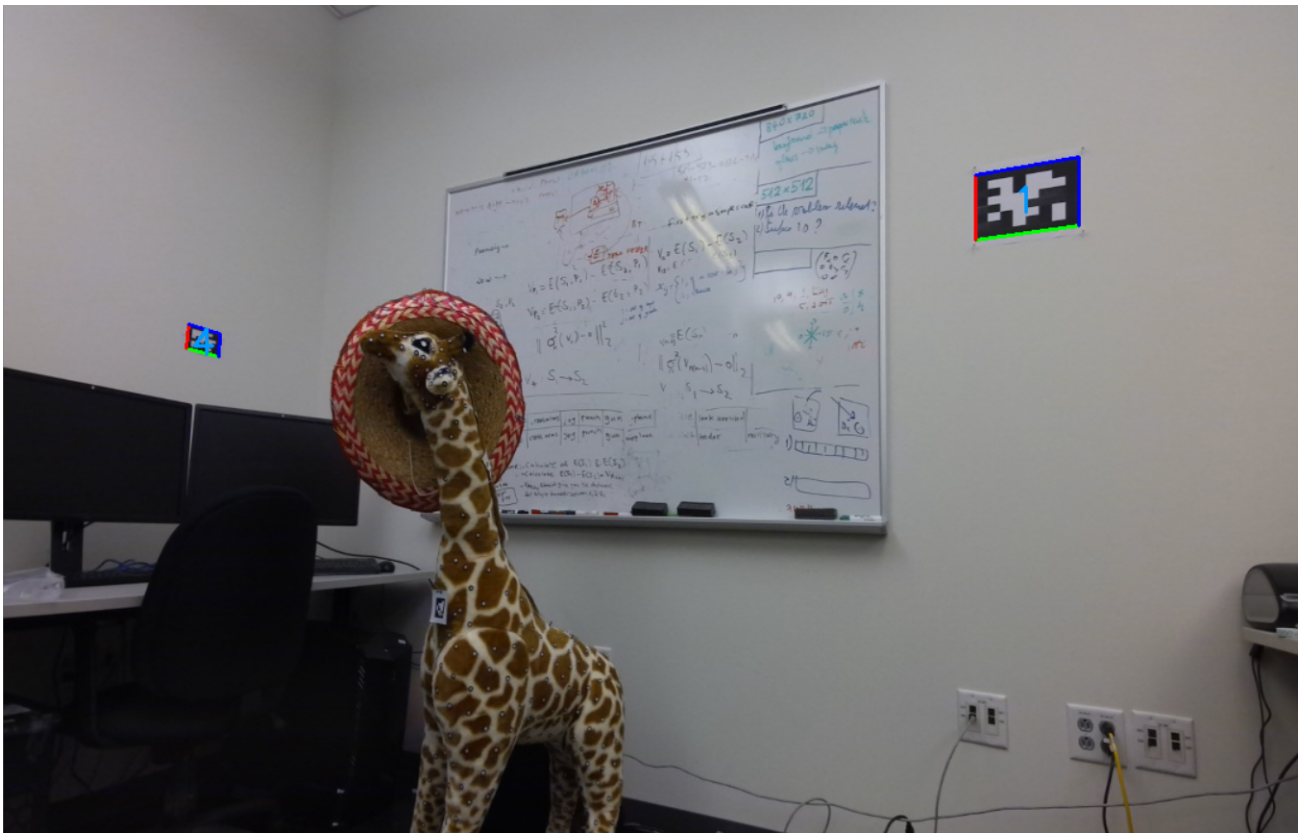


Figure 4.15: AprilTag marker detection in 2d

point of the 2D marker points(pixels). But as the resolution of depth and color camera are different, not all the color pixel will have its corresponding depth point present in the point cloud. Therefore, we interpolate the corresponding 3D point in point cloud. First step is to 3D reconstruct our point cloud thus converting it to triangular meshes. As our point cloud is organized, we pick points from neighbouring pixels as candidate mesh neighbours. But because, some points may be invalid due to sensor error, we check if those points are valid first. After that we check the distance of neighbour points from interest point whether the distance is less that the distance threshold which is 3cm. We also check angles of the triangle. We do not accept less than 15° angle between two sides. Finally, we remove all the points which does not belong to any faces.

4.4.4 Projection of 2D Point to 3D

We take the following steps to project from 2D pixel point of color camera to 3D point in Depth camera:

- After creating mesh from organized point cloud, we project 3D meshes into 2D meshes where in 2D mesh, vertices are pixel points in color image. While doing so, we make sure the pixels of color images have same index in 2D point vertices as 3D points have in 3D point vertices. We have talked about the mapping 3D point to 2D pixel of color image in 4.1 .

- After creating 2D mesh, we employ a Hash Mapping algorithm and thus create a $n \times n$ grid where four corners of the grid represent four extremes of the 2D points in the 2D mesh. Each cell of the grid contains a bucket of 2D faces. While hashing a 2D face into a bucket, we determine the four extremes of the face and hash that face into all the buckets covering the range set by those four extremes.

- While each hash operation has constant time complexity, the average complexity for total insertion operation for each face is θ(B) where B is the average number of buckets on which each face is hashed. On the other hand, search operation is O(F) where F is the average number of faces in each bucket. That means, if the number of buckets is low then our insertion cost decreases and search operation increases. On the other hand, if it is higher then insertion

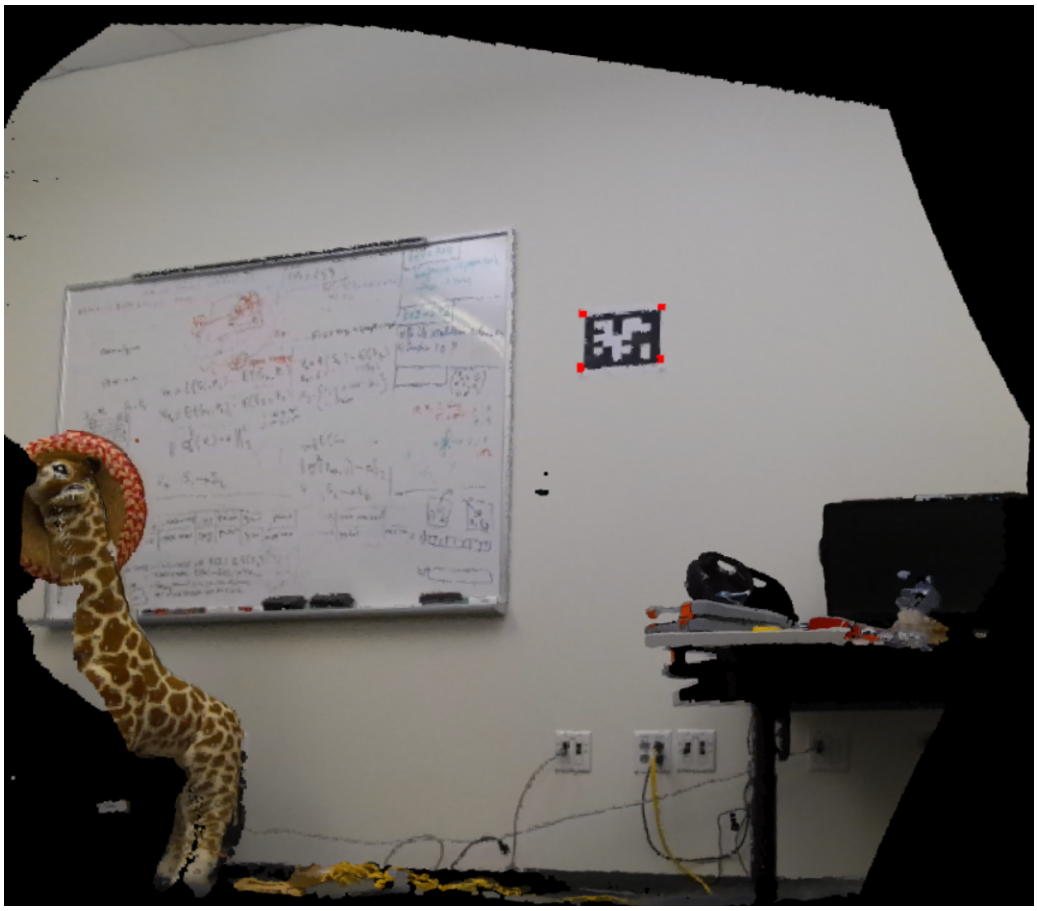


Figure 4.16: AprilTag marker detection in 3d

operation increases and search operation decreases. Because, the number of markers is very sparse compared to the number of points present in the images, naturally insertion operation is much costlier compared to search and retrieval operation. Due to this reason we make the size of each bucket large so that insertion operation takes less time. In other words, we keep size of n low.

- For the search query, we hash each marker and find its corresponding bucket in the grid. We extract all the faces from the grid and check which face the marker belongs to.
- After getting a match, we compute the barycentric coordinate of the pixel point with respect to the 2D face it belongs to.
- Using this barycentric coordinate and corresponding 3D face, we finally interpolate the 3D point in the point cloud.

4.4.5 Normal Computation

While using markers (AprilTag) for 3D registration normals are not needed. But, to give live feedback to the user helping to align the camera in a better orientation, normals are calculated in real time. To achieve the efficiency, we created the point cloud as organized set of points. Therefore, unlike unorganized point cloud we can search for neighbouring points by taking account their pixel position with respect to the pixel position of the interest point.

We use an average based surface normal method (Holzer et al., 2012), where we smoothen surface based on depth of the point and change of depth with respect to its neighbours. Because points with larger depth has worse noise to signal ratio, the smoothing window is larger for points with larger depth. On the contrary, points with lower depth value has lower smoothing window. To smoothen surface efficiently, we use integral image to compute the average of the window faster. Finally, we take cross-product of the neighbouring four points to get the surface normal. Tough this method is not highly accurate near object border; it is highly efficient achieving our real time normal computation goal. After calculating the normals, we calculate the angle between the camera z axis and the normals of the surface. We take average of the normals from the middle part of the

Figure 4.17: Surface normals calculated on real time (Point Cloud)

cloud, middle 540×440 of 640×540 as measurement within middle part of points has less error than the side ones.

4.4.6 Pose Selection

We provide visual aid to user to help them to choose camera pose wisely. We have designed a GUI to facilitate this purpose. The GUI is made with QT framework (Fig: 4.18). This GUI consists of four panels. Upper left panel contains options for the user to add a frame to reconstruction process, remove a frame from the scene and save current reconstructed model. The rest of the panels are used for giving visual aid to the user. At upper right panel, user sees a color frame of the current pose of the scene. Markers which are successfully projected into the point cloud is shown with yellow, blue, green and red square block at their four corners. Each little square block represents a feature point. If the marker is visible in the RGB frame but not in-depth frame, we cannot use that marker. User recognizes such markers by observing a large red block in the middle of the marker. If the current pose is optimum with respect to the presence of markers in the scene

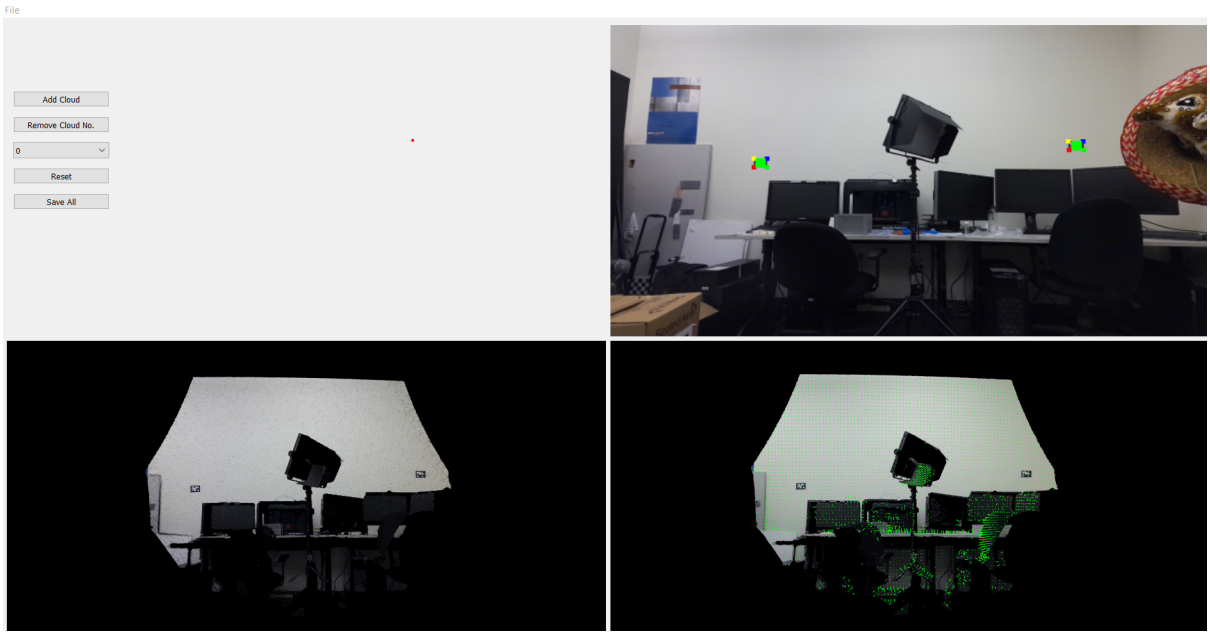


Figure 4.18: Graphical User Interface for Visual Aid

Figure 4.19: Screen turns green if there are markers in the scene and camera pose is within optimum position with respect to the scene (RGB image)

and around 75% of points orientations with respected to the camera are within the optimal angle range of 60°to 120°, this panel turns green (Fig: 4.19). This is a signal to user that the current frame is a good fit for the reconstruction. We have arrived at this hyper-parameter (suitable camera pose) by the experiment done in sec: 4.2.1. At lower right panel we show a point cloud demonstrating real time surface normals (Fig: 4.17). Upon taking visual aid, user can select the pose. After selection, updated global scene immediately appears on the screen at the lower left panel. Moreover, if the user does not like new scene, they can choose to delete the last frame and continue from there.

4.4.7 Global Alignment

After user adds a new frame, we calculate the 3D correspondence of markers in the current scene. We check which of these markers are already in the global scene. The calculation for the correspondence for each pair of marker happens in constant time O (1), as each marker has a unique id and we compute the 3d position of each 2d marker at the time of taking the pose. We find the correspondence with common markers in global scene and using the algorithm mentioned in 2.4.1, we compute the transformation matrix of the current pose with respect to the global scene and align it with the global scene. Also, we add new markers which were not previously in the global scene.

4.5 Implementation Details

The programming language used for both systems was C++. For image acquisition (depth and color image), Kinect API was used. Also, to generate the 3D point cloud Kinect API used. In case of marker-based system, to detect AprilTag, we used AprilTag library available from the author's website. As the program is online, to make it efficient multi threading was used. There are three main thread. One is to acquire images with a frame rate of 30 milliseconds. Second thread was used to detect the markers in the image and aligning the image globally. The third, thread was used for interacting with aligned images and the current images in the GUI. For multi threading and GUI, QT framework was used. For visualization in the GUI, PCL was used. Since, there is no official release available for using PCL with QT, we had to build it ourselves. Apart from that, rest of the coding was done by us. We tried to make marker-based system as independent as possible. Given,

Table 4.4: Summary of API and Libraries used in our systems

API / Library	Version
Azure Kinect	1.4
OpenCV	4.1.1
QT	5.13
PCL	1.10

camera intrinsic, extrinsic and depth table users can use this system with other RGB-D cameras also.

For the feature-based system, we used Kinect API and PCL. Here is the summary of the libraries used in both of our system in Table: 4.4

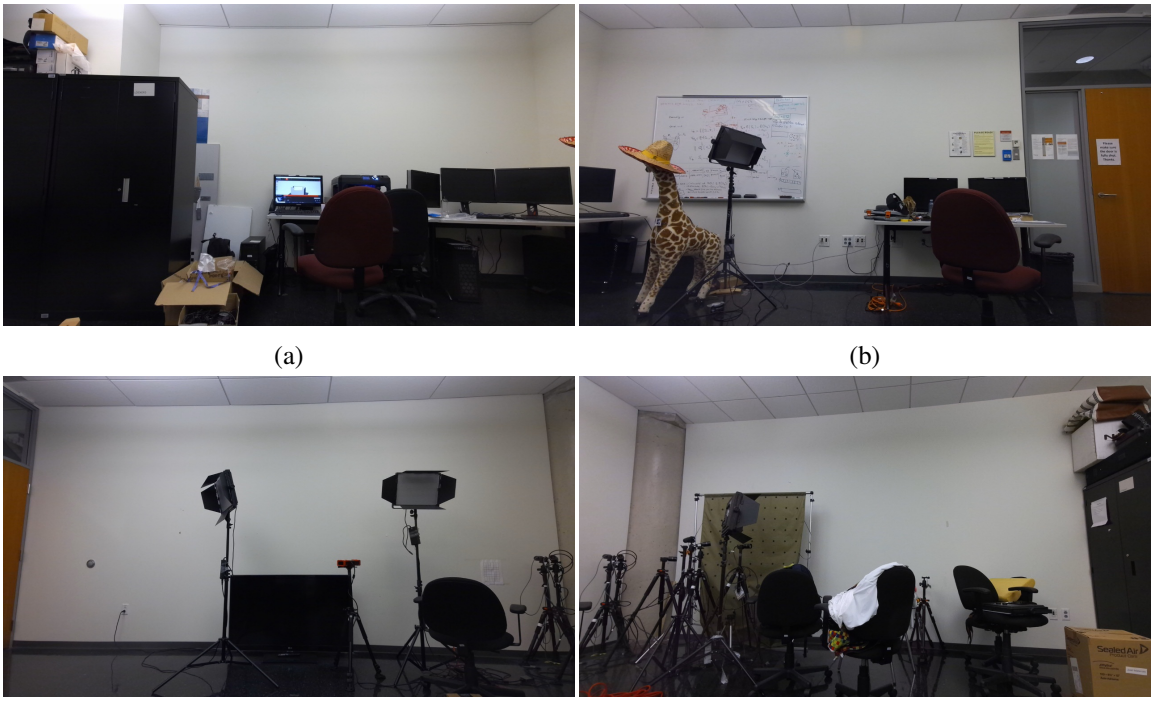
Chapter 5

Result And Analysis

In this section we discuss about the registration of point clouds obtained from two of our developed systems, real time marker-based registration and features based offline registration. We show our 3D reconstruction results through various scenarios and compare both systems through empirical means and visualization.

5.1 Dataset

For the experiments we acquired images from two rooms. One room is a laboratory situated at Concordia University and the other one is a bedroom from a residential apartment. These two rooms provide different kinds of geometry and texture which is useful to test both of our system. On the other hand, the common thing about these two rooms is that, among the four walls, one is lacking in both texture and geometry.



(a) (b)

(c) (d)

Figure 5.1: Laboratory environment

The laboratory is an almost a square room. The dimension of the room was measured by a tape. The length of this room is 6600 mm and width are 6510 mm. It mostly contains one whiteboard, one doll of giraffe, computers, monitors, chairs, tables, cameras, camera tripods and light stands. Most of the objects in the laboratory is of rectangular shape. Therefore, there are not enough diversity in terms of geometry. Also, most of the objects are either white or black in color. So, there is not much texture present in the environment either. From 5.1, we can see in (c) that this side of the room lacks in geometry and specially texture. One positive aspect of the laboratory though there is no variance of lighting in the environment which is very helpful while acquiring images with RGB-D cameras like Kinect.

(a) (b)

(c) (d)

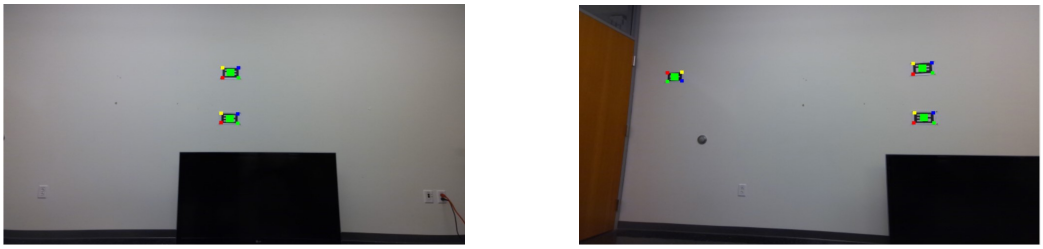
Figure 5.2: Residential environment

The residential room is rectangular. Its width is very short, about four meters which makes it difficult to take images with Kinect from suitable distance as Kinect has a depth range. The bedroom contrasts the laboratory in terms of texture and geometry. It contains one table, one chair, one bed, desktop, monitor and other household objects. There is verity of color present in this dataset and the geometry also is diverse. One problem of this dataset is though, one of the wall (b) in 5.2 lacks in texture and geometry.

5.2 Marker Based Real Time Registration

Marker based registration pipeline is an online system with feedback system to aide user in registration process. The feedback process was discussed in methodology. Now, we talk about the image acquisition results. In case of detecting markers in depth image, result was very accurate while acquiring images from forward (within thee range 60°- 120°with respect to the image plane) as shown in Fig: 5.4 with average error being only 1.2 mm. The total error is calculated by summing

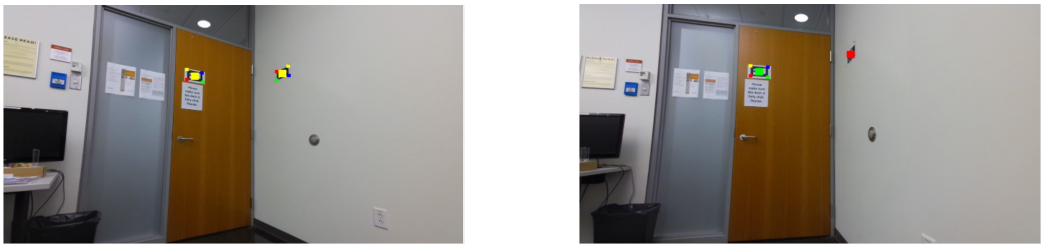
up the distance between a corresponding marker pair. Then total error is divided by the number of markers available in the scene.



(a) (b)

Figure 5.3: Image acquisition within 60°- 120°. Green marks on tag means all the points were successfully projected to depth image from color image.

On the other hand, images taken outside of the range (60°- 120°) was difficult to attain all the markers.

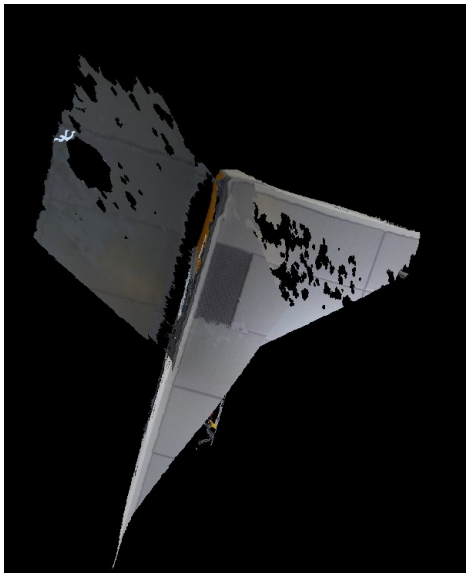


(a) (b)

Figure 5.4: Image acquisition outside the range60°- 120°. Yellow marks on tag means three of the points out of four were successfully projected to depth image from color image. While red marks mean two or less points out of four could not be projected to depth image.

Even though some of the markers points could not be detected in RGB-D image, we still could register images without any difficulty as shown in fig: 5.5

(a)



(b)

Figure 5.5: Successful registration with very few marker point in a corner case scenario

Moreover, due to high robustness of the markers, they are very reliable. Which enables not only to use very few markers but also register the images with least amount of overlap as shown in Fig:5.6 and Fig:5.7. We were able to register half of the lab room with only two tags with high accuracy.

Figure 5.6: Highly accurate registration with least amount of overlapping with only two tags (side view)

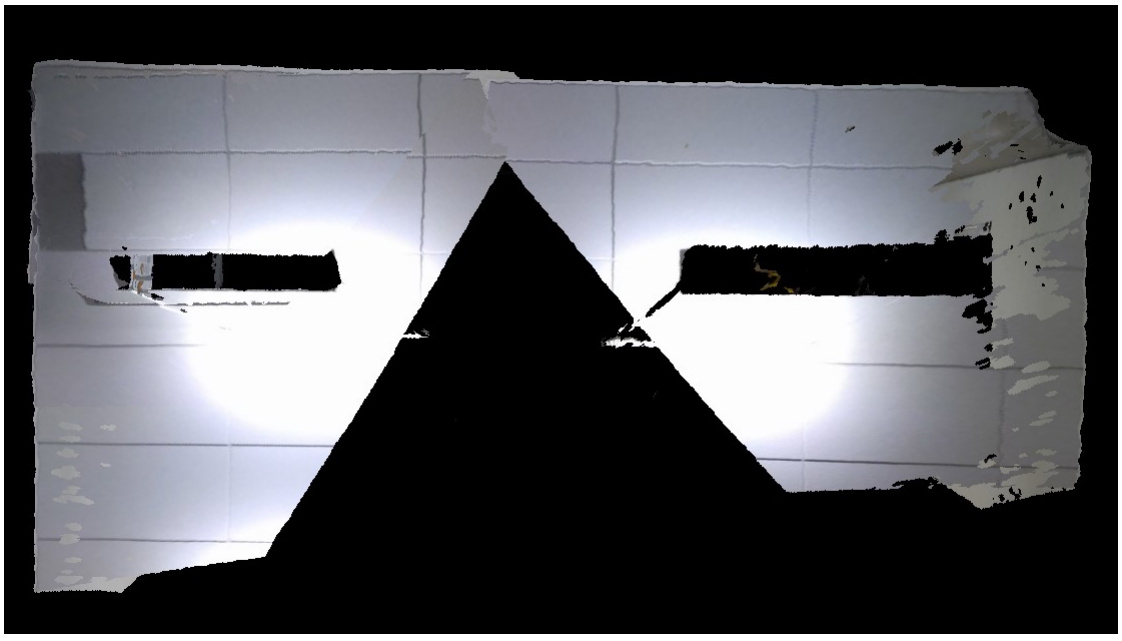
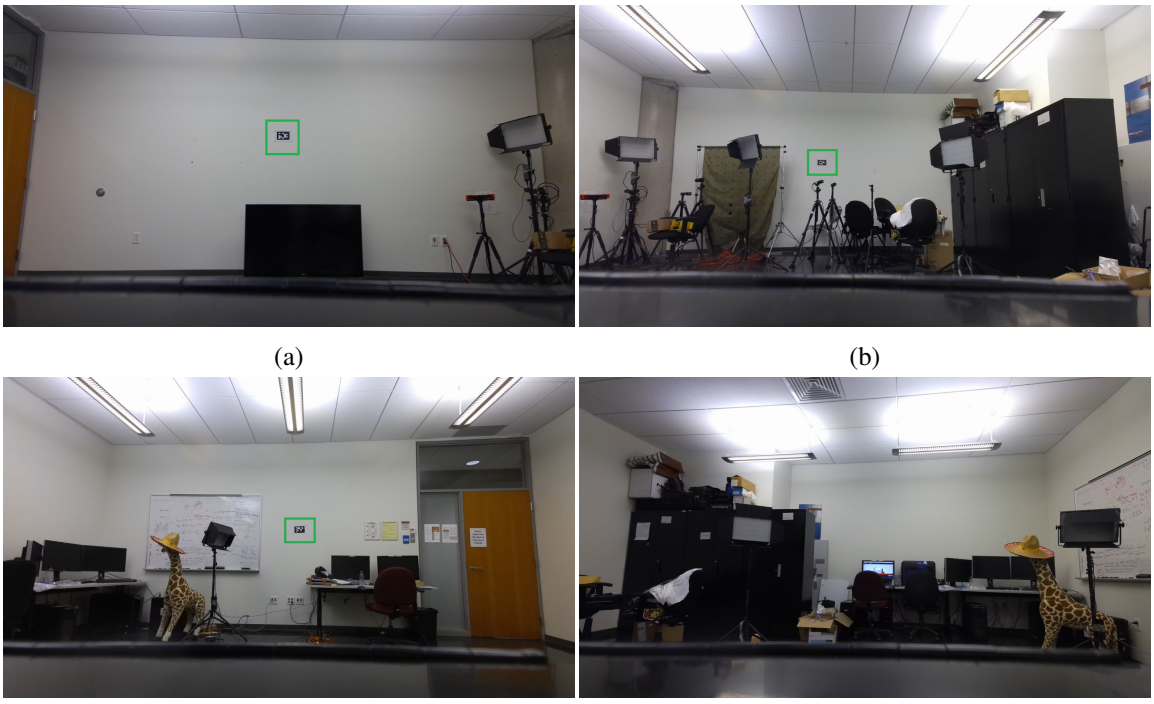


Figure 5.7: Highly accurate registration with least amount of overlapping with only two tags (top view)

The lowest number of AprilTag that would be needed to register the lab room with the resolution

of Kinect would be 3. We experimented if our system can achieve such result. To test this we used only three AprilTag on three walls of the lab room. We placed each AprilTag at the middle of the room. One wall was left blank. From Fig:5.8, we can see that walls (a), (b) and (c) has one AprilTag each and (d) does not have any.



(a) (b)

(c) (d)

Figure 5.8: Laboratory Room Registration With Three AprilTag

During experimentation, we observed we could successfully register the entire room with only three AprilTag while only taking four poses as shown in Fig: 5.9 and Fig: 5.10

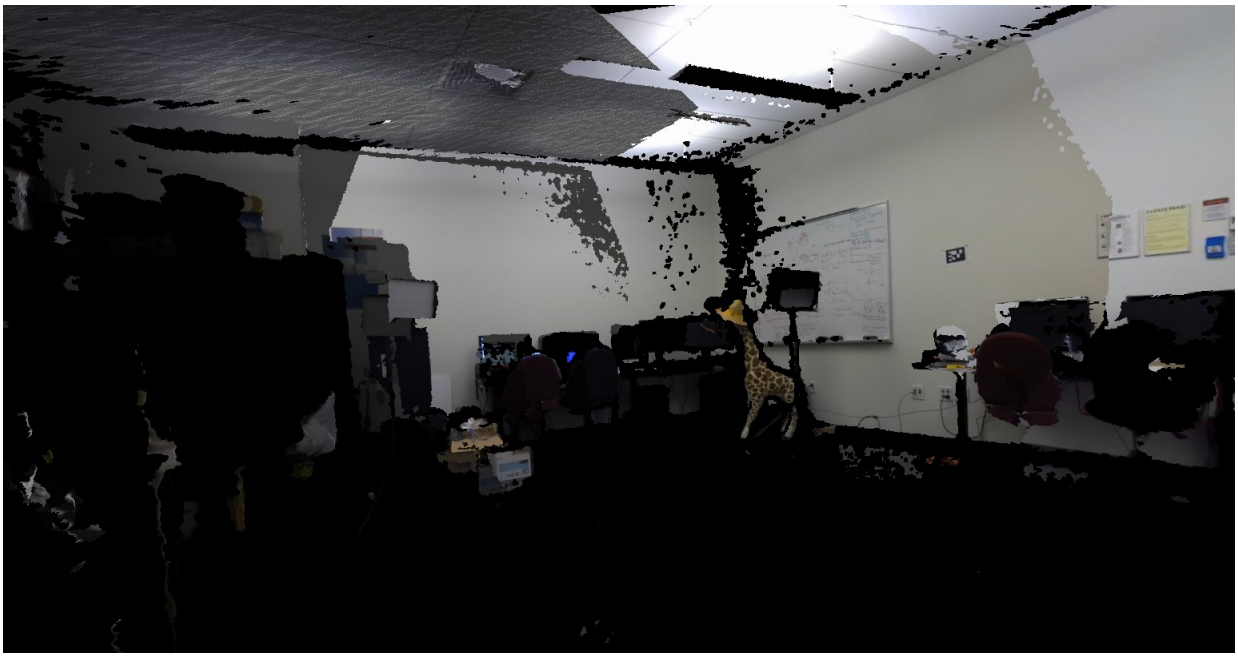


Figure 5.9: Global registration of the lab room with three AprilTag (side view)

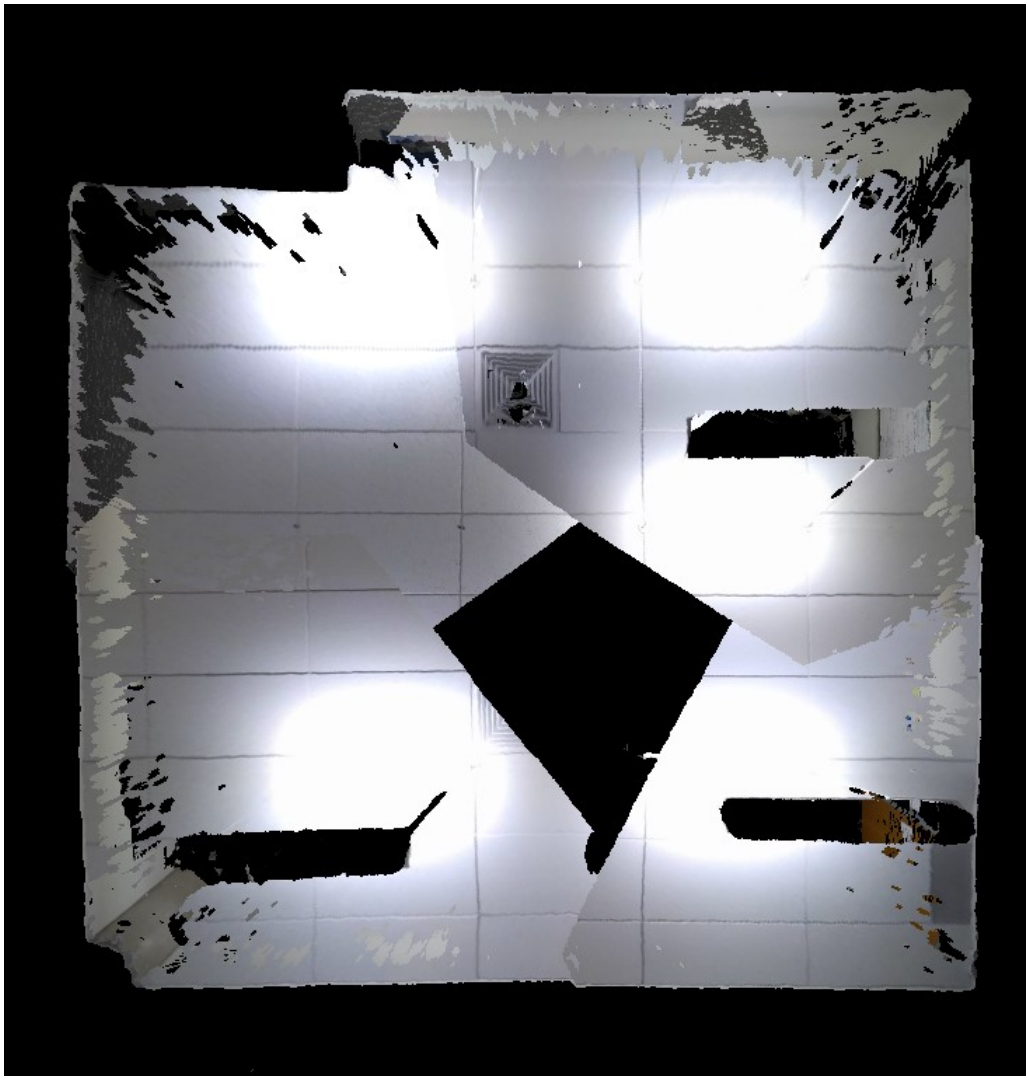


Figure 5.10: Global registration of the lab room with three AprilTag (top view)

Although, the registration was successful, we can observe that the point cloud is sparse, and the alignment is not perfect. To obtain a dense point cloud and to test our system can handle plenty of markers we placed 26 AprilTag in the lab room again and recommenced the experimentation. We took total 10 poses. Finally, even after the closing loop while taking images, our system produces no noticeable drift error as shown in Fig: 5.11. Although we have noticed while in all the previous alignments the average errors remained below 4 mm, at the last alignment of loop closing the average error increases to 7 mm for last pair as shown in graph 5.13 .But this error still too small to have any noticeable impact. Therefore, this system does not need any post processing to close the loop.

Also, compared to the previous alignments with only three AprilTag, this point cloud is denser as shown in Fig: 5.12.

Figure 5.11: Dense Global Registration of the Lab Room (top view)

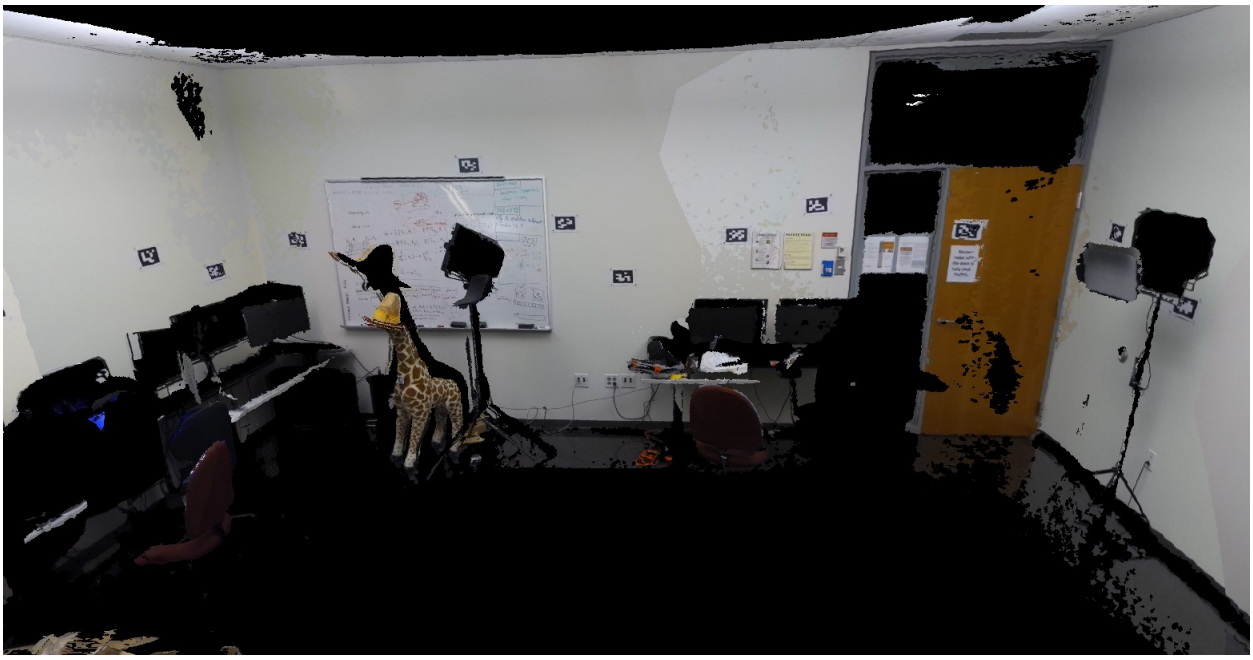


Figure 5.12: Dense Global Registration of the Lab Room (side view)

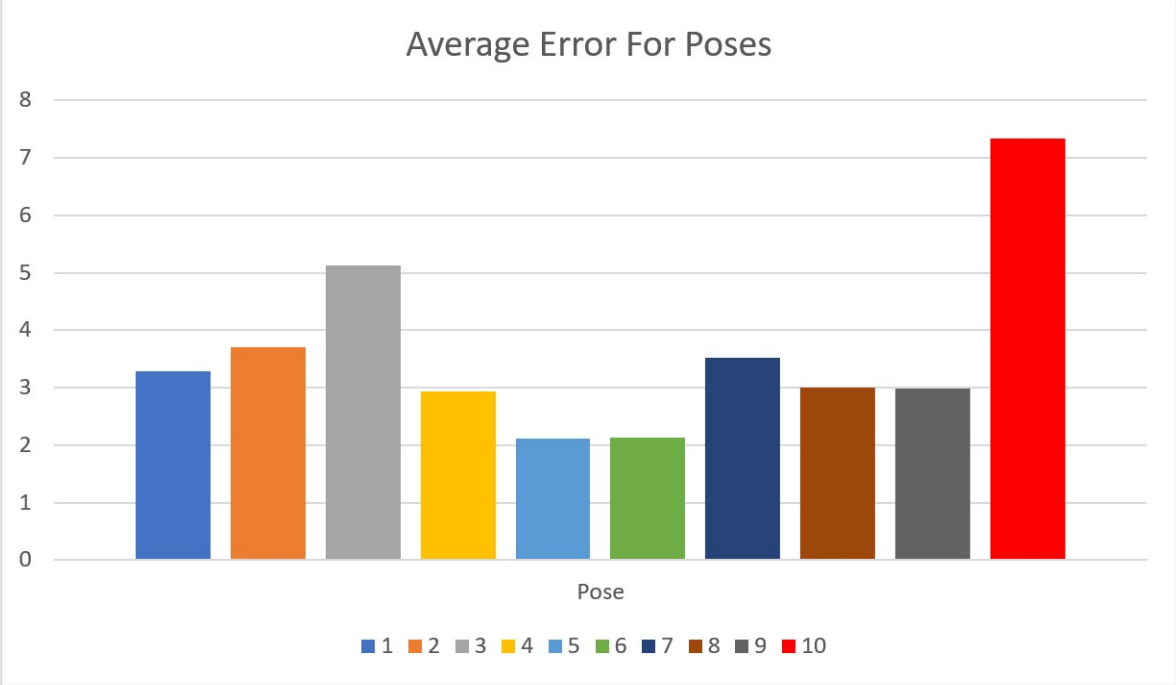


Figure 5.13: Average Correspondence Error for each poses

We also ran same experiment in the residential room. Because this room is much smaller compared to laboratory room, we placed 12 AprilTag and took 8 poses and acquired similar result as in Fig: 5.14 and Fig: 5.15. Also, the pattern of average correspondence is similar that is, while most of the average errors are below 5, the error for last errors increases to around 8 mm as shown in Fig: 5.16. We saw a slight increase in error in this case. We attribute this error to the lighting condition of the room which was not perfect for AprilTag.

Figure 5.14: Aligned poses during capturing images

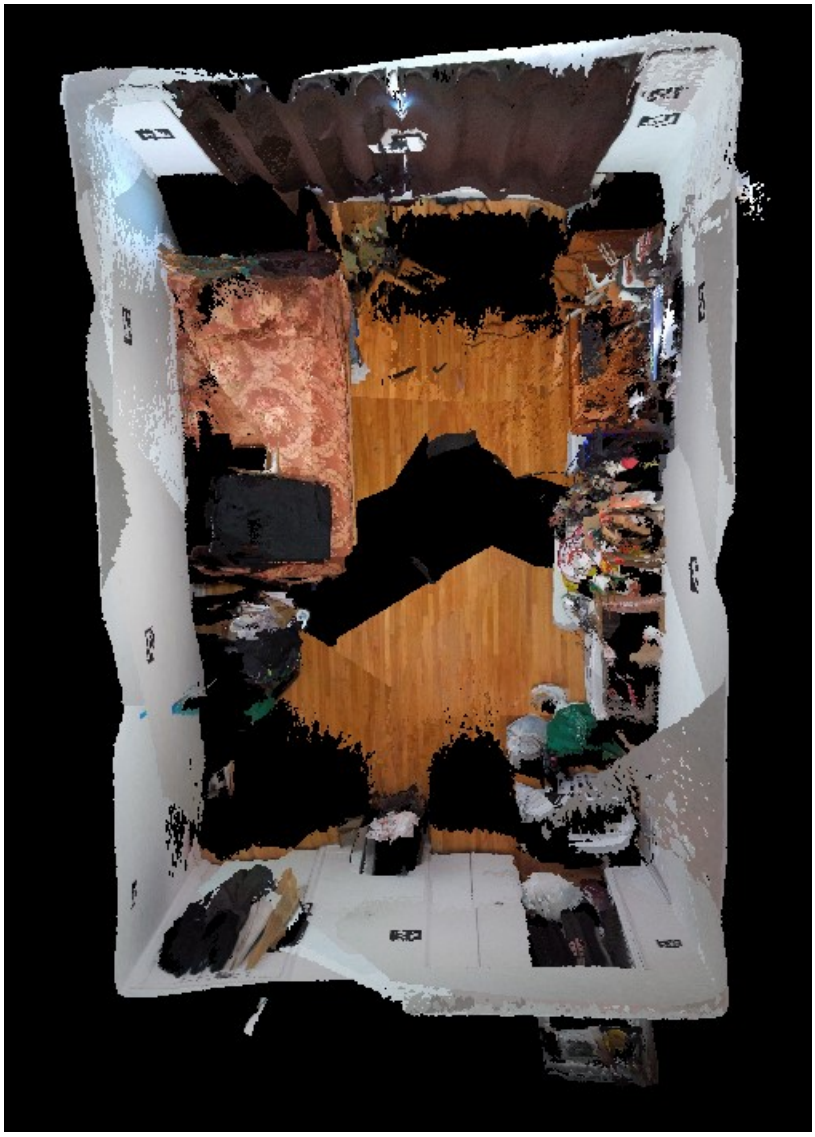


Figure 5.15: Global registration of the residential room (top view)

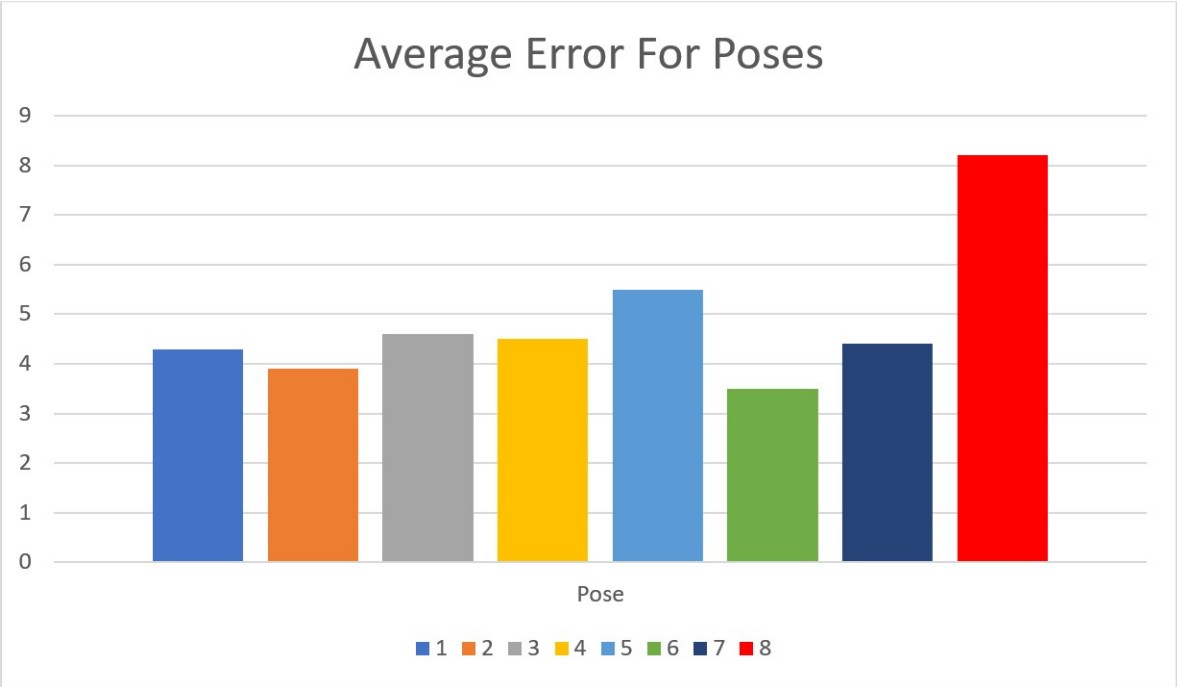


Figure 5.16: Average Correspondence Error for each poses taken in residential environment

5.3 Feature Based Registration

We showcase the registration result of feature-based system and its accuracy on scenes with fair number of features and low number of features. For image acquisition, at first, we took images without markers. We took images of the environment where features are scarce. Then, we tried pairwise registration on the images. We show that registration does not work well on such places. For example on laboratory dataset, at 9 degree rotation difference, we were capable of registering images in most of the cases but, the pair wise registration of images with scarce features gives poor result for (c) part of Fig: 5.1.

Figure 5.17: Incorrect pairwise registration at 9 Degree of difference having less features

From Fig: 5.17, the red rectangular area denotes that the pairs did not align well.

Figure 5.18: Pairwise registration two images at 9 degree of rotation difference with AprilTag

To demonstrate the failure of registration is only due to the lack of features, we used AprilTag in the same environment and could successfully register the same pose as in Fig: 5.18. Although, we used AprilTag to aid in registration process in this experiment, we did not use them as markers rather they just act as if they were natural feature present in the environment. Of course, same feature-based pipeline is used for both cases. From the images we can see that, while enough features are present, this pipeline works well in case of pair wise registration.

Figure 5.19: Pairwise registration at 6 degree of rotation with low features (pose 30-31)

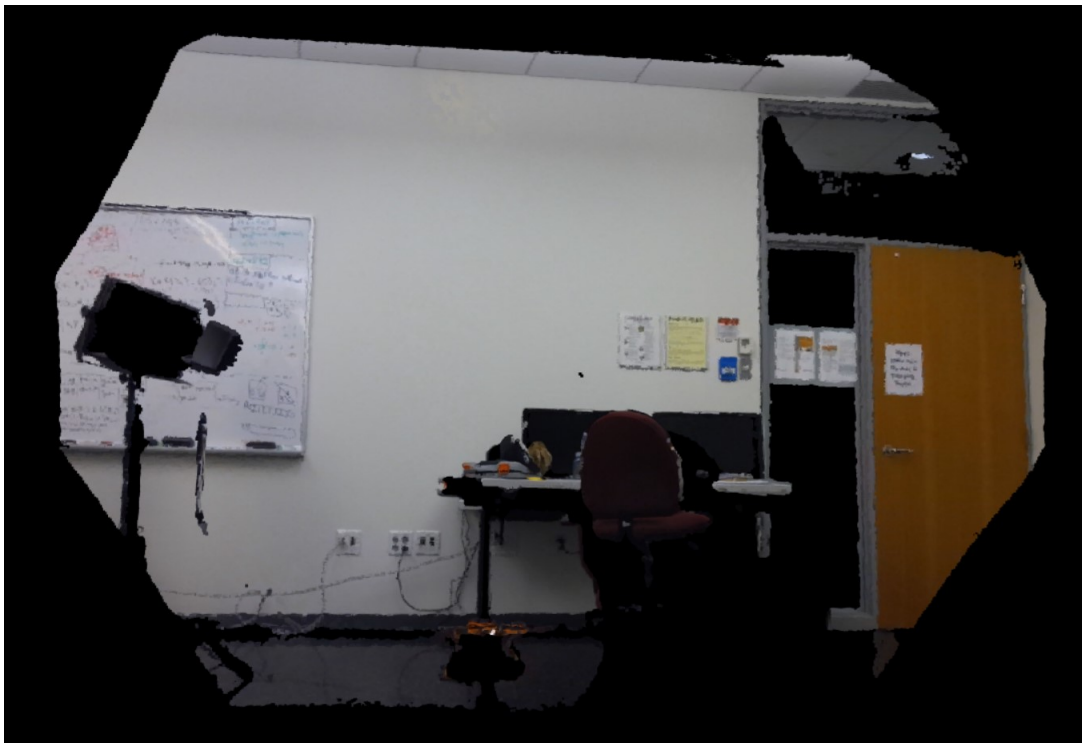


Figure 5.20: Pairwise registration two images from middle of the scene (pose 17-18)

However, at 6°of overlap, we were able to do pairwise registration without AprilTag even when

there is a lack of features in the scene as can be seen in 5.19 and 5.20.Therefore, we maintained 6°of rotational difference between two consecutive poses and for the entire room we took 60 poses.

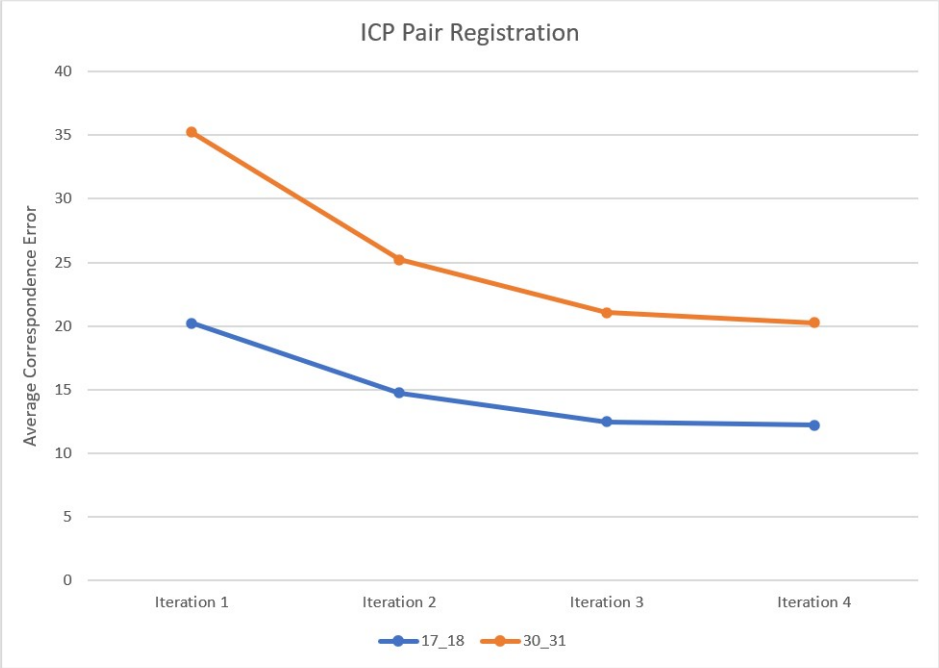


Figure 5.21: ICP iterations for 5.19 and 5.20

Although the point correspondences are slightly more erroneous and it takes more iteration to successfully register in case of scene with low feature than fair amount of features as can be seen in 5.21. However, in terms of visual there is no apparent difference.

Corner cases are particularly challenging while doing registration due to images taken from angles which are not optimal for accurate measurement, it does work well in these kind of scenario too (5.22 - 5.29).

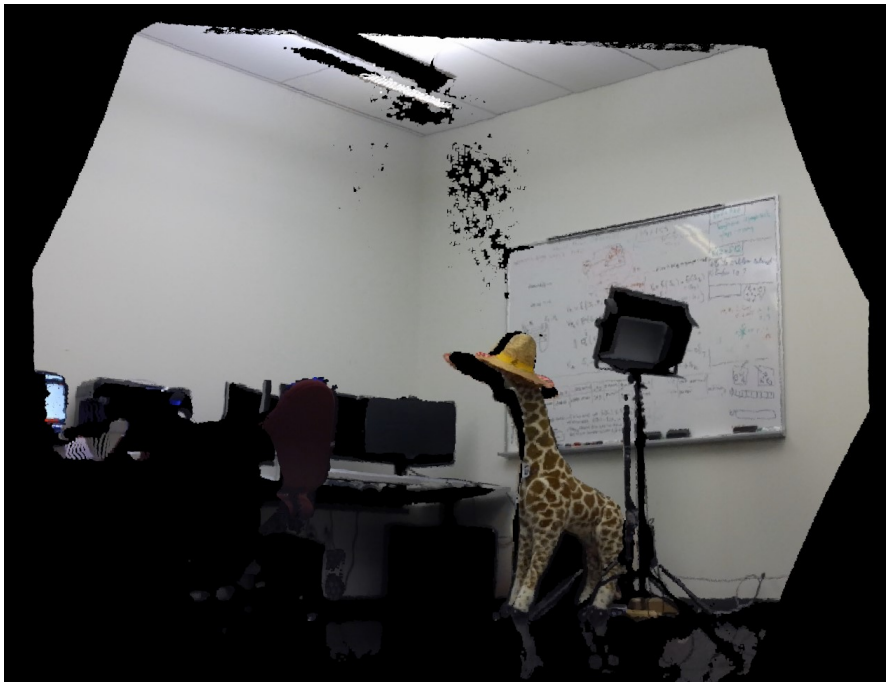


Figure 5.22: Registration of images taken from first corner(side) of the scene

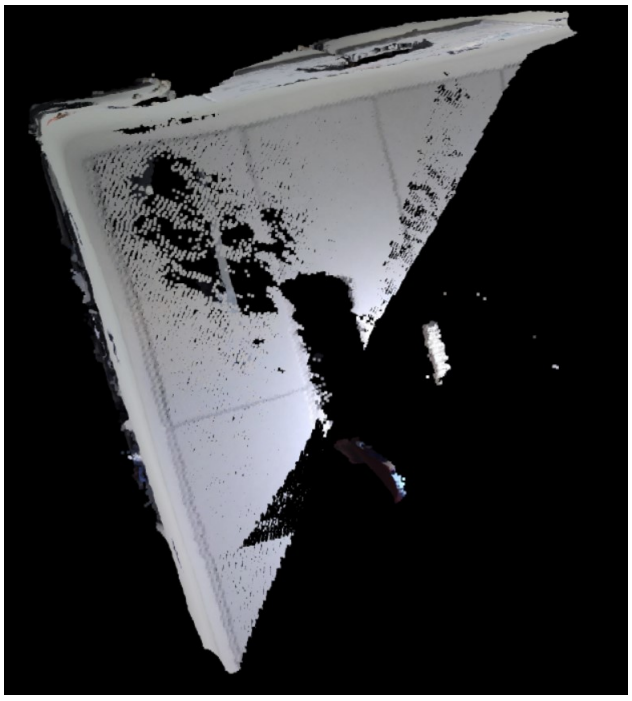


Figure 5.23: Registration of images taken from first corner(Top) of the scene

Figure 5.24: Registration of images taken from second corner(Side) of the scene

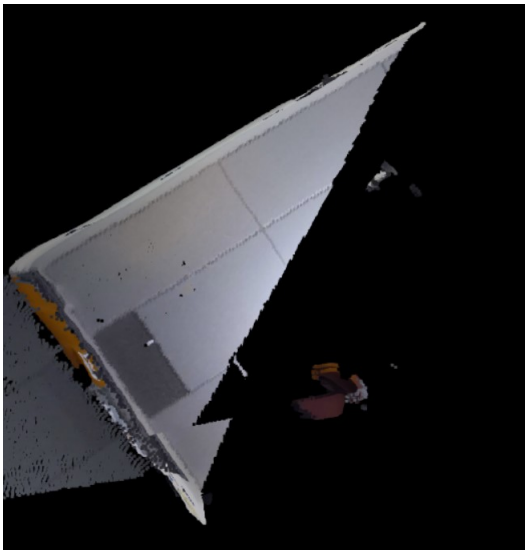


Figure 5.25: Registration of images taken from second corner(Top) of the scene

Figure 5.26: Registration of images taken from third corner(Side) of the scene

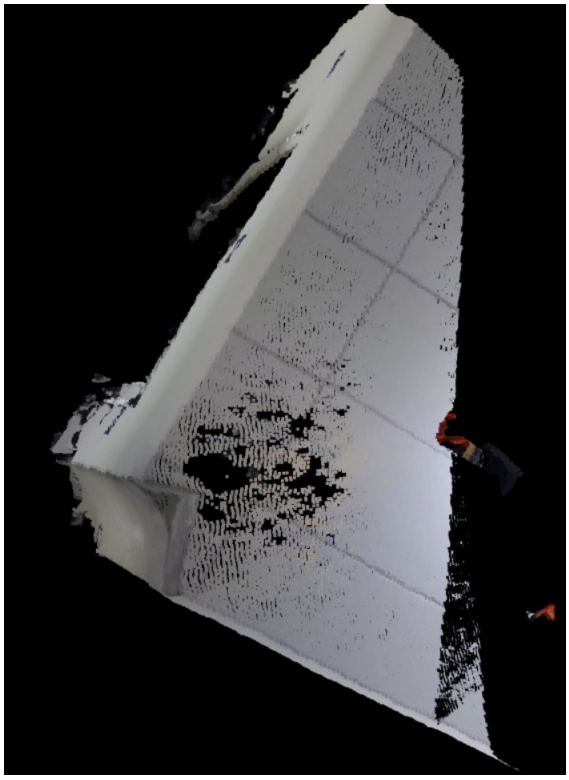


Figure 5.27: Registration of images taken from third corner(Top) of the scene

Figure 5.28: Registration of images taken from fourth corner(Side) of the scene

Figure 5.29: Registration of images taken from fourth corner(Top) of the scene

While we have achieved accurate pair wise registration, there are still some small errors remain.

In time of global registration these errors add up and create drift as shown in 5.30 .

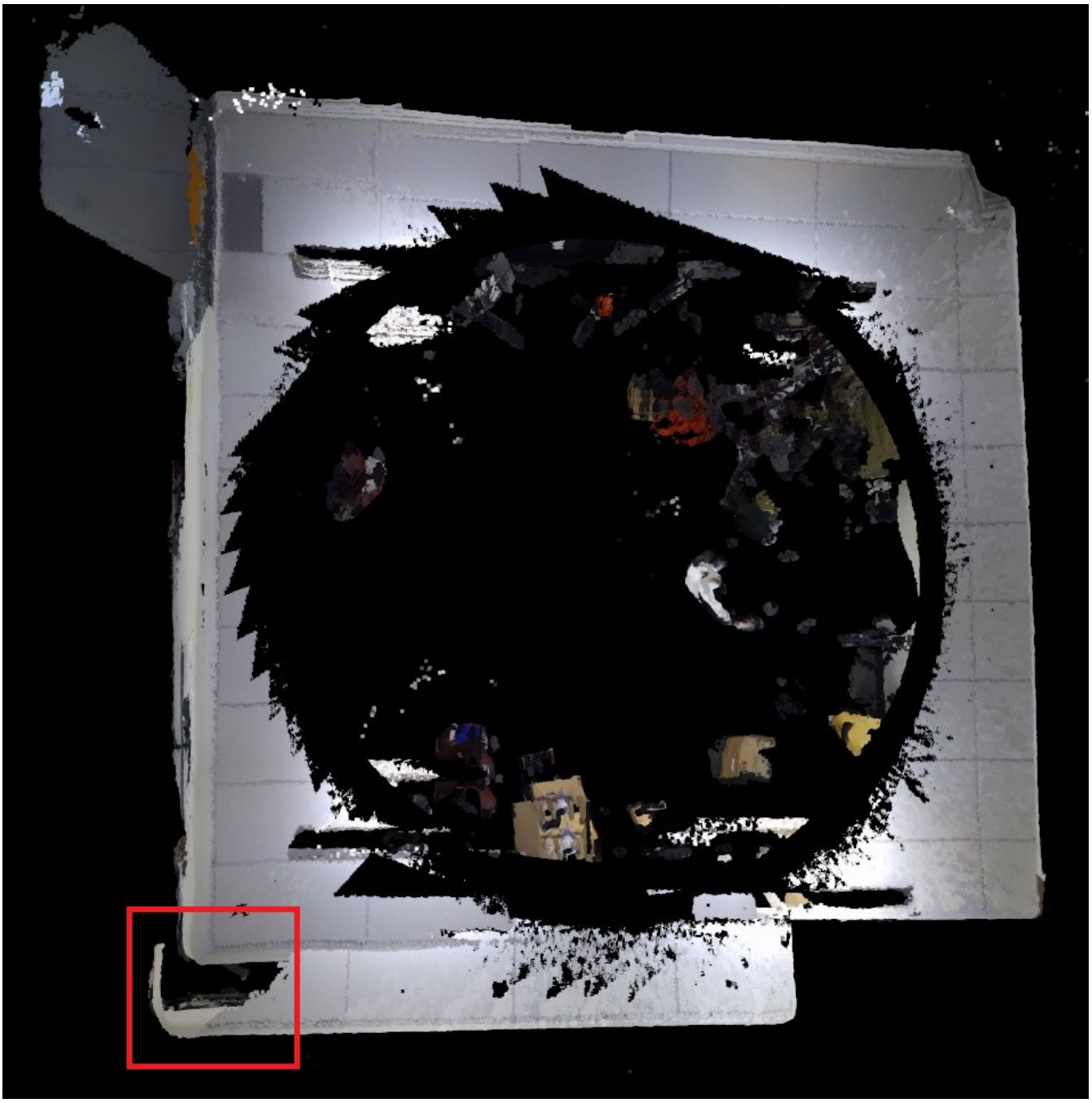


Figure 5.30: Drift after pair wise registration

Therefore, an ICP based global registration is also performed to reduce the drift error and close the loop as in 5.31.

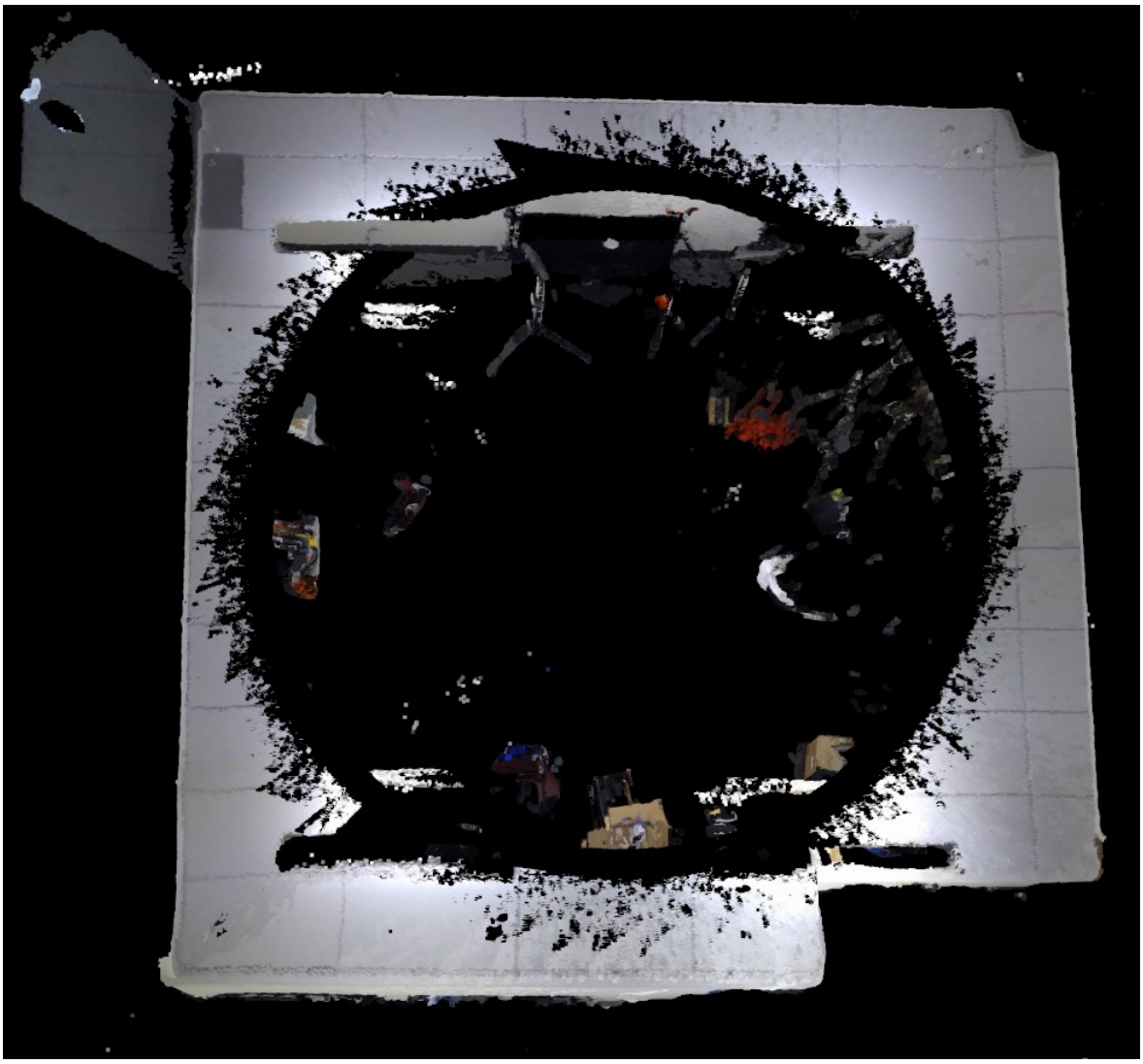


Figure 5.31: ICP Global Registration

From the 60 poses that were aligned consecutively, we have noticed the average correspondence errors for features remain mostly in the range of 10 to 20 mm. The lowest error was 7.2 mm and the highest was 29.54. From Fig: 5.32 we can see that the last pose records highest amount of error. This is due to the fact that, those three frames involves in closing the loop and thus have high amount of error.

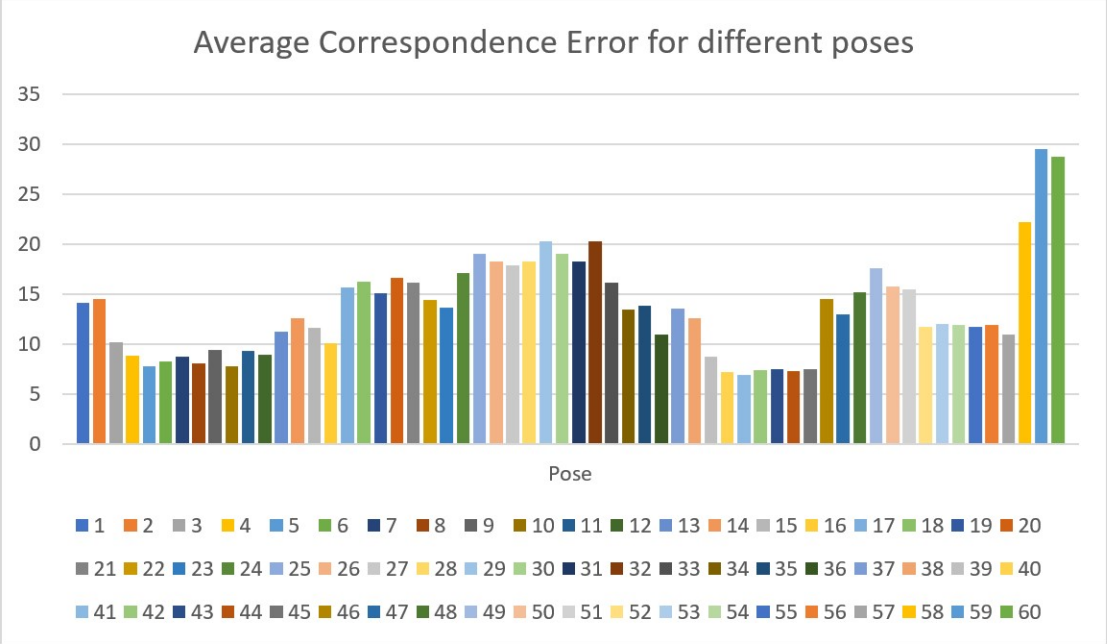


Figure 5.32: Average correspondence error for each poses taken in laboratory room

We follow the same procedure in case of residential dataset too. After performing pair alignments and initial global alignment, we observe the drift error here too.

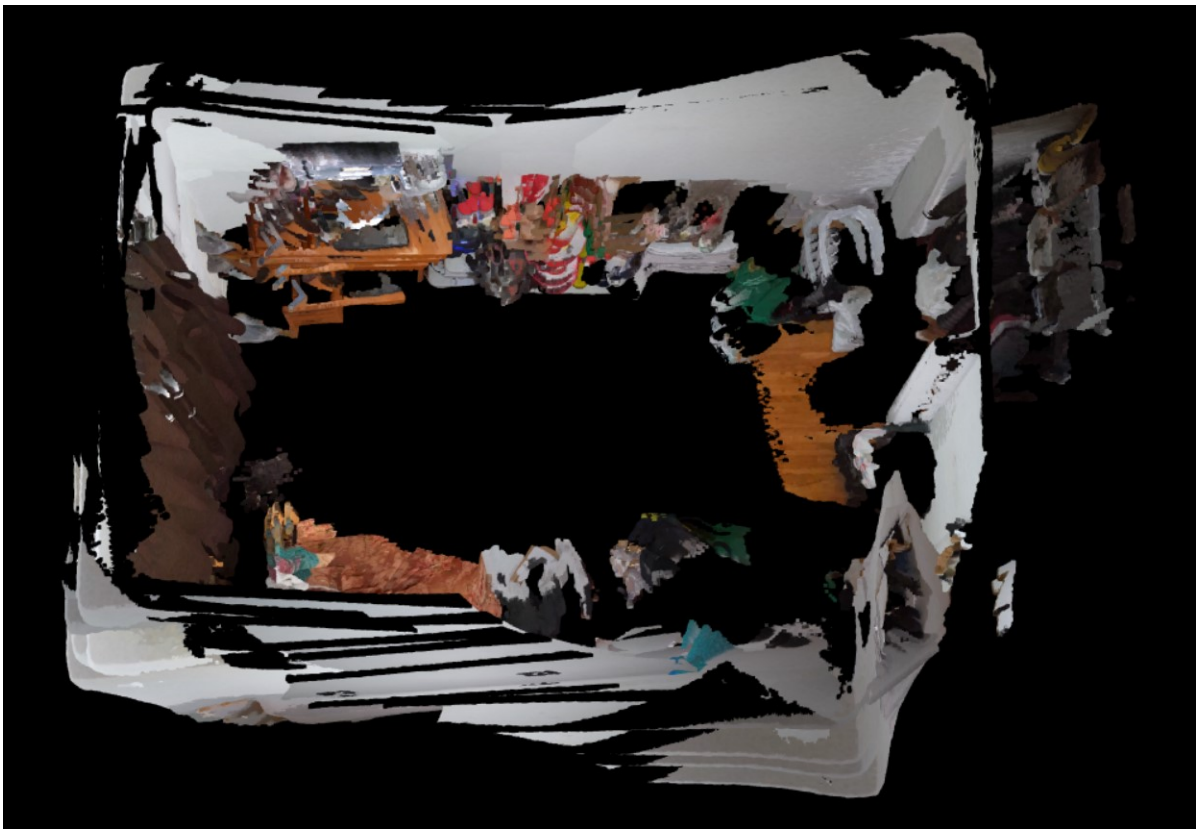


Figure 5.33: Drift error in initial global alignment in residential dataset

Finally we perform global registration to remove drift error as shown in 5.34. The error patterns as shown in Fig: 5.35 are same as poses taken in laboratory room. For most of the poses the average correspondence error ranges from 10 to 15. But there are two peaks. The last peak is understandable as they are used to close the lab. The poses responsible for middle peak are taken from the third wall, image (c) in Fig: 5.2. We can see that these poses consist mainly a curtain of same dark maroon color. Also, at the right corner there is windows which is closed by glass. As glasses are reflective, that may also contribute to the error as TOF cameras are not accurate with glass objects.

Figure 5.34: Global registration of residential dataset

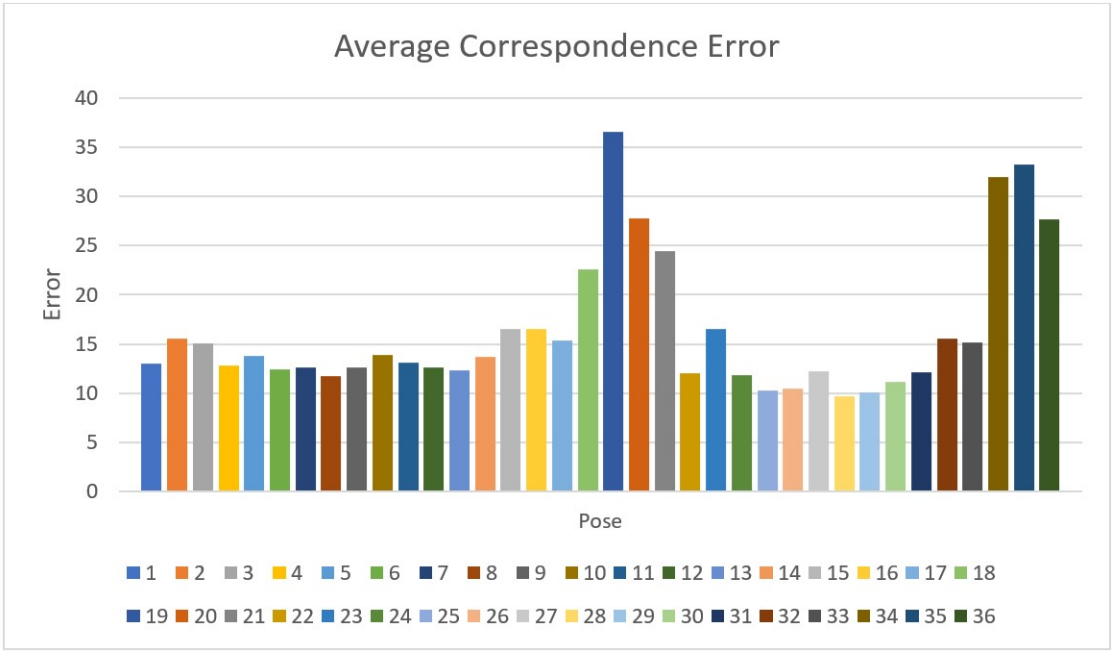


Figure 5.35: Average correspondence error for each poses taken in residential room

5.4 Result Comparison

In comparison of the two systems that we have presented, marker-based registration system showed more accuracy with least amount poses taken. Also, it is done online. While feature based system also showed good alignment, it took 15 times more poses compared to sparse marker-based registration and 10 times more compared to dense marker-based registration. It is important to note we are not using any bundle adjustment process in case of marker-based method to attain efficient online alignment in contrast to offline methods where bundle adjustment is a necessity. In case of offline methods, many pictures are taken which in turn accumulates greater drift. But due to bundle adjustment those drift errors can be corrected. On the contrary, drift errors impact badly in case of online systems as bundle adjustment is not being used. Therefore, in case of online methods, our goal is to take a smaller number of images which in turn should accumulates less drift error.

Also, this system was aligned in offline mode which takes fair amount of time. It may seem down-sampling can increase the speed. But there are two problems. One, down-sampling makes the cloud UN-organized. When that happens, neighbouring pixels are not necessarily actually neighbours of the points. But we need to find the nearest neighbours while calculating features, doing ICP or using RANSAC. Finding nearest neighbours in unorganized point cloud is a costly operation. Two, heavily down-sampling decrease the quality of the features as feature calculation needs nearest neighbours as due to down-sampling, the radius for neighbours must be increased compared to prior to down-sampling. That makes the influence of farther points in calculating features greater than before which contributes to feature ambiguity. Also, feature based system needs parameters fine tuning. Depending on the point cloud, feature estimation radius, RANSAC rejection radius, feature matching threshold, number of Iteration, etc. needs to be tuned in order to achieve a good registration. On the other hand, our marker-based system is free from all of these problems.

To measure the accuracy of the alignment of both systems, we also measured the dimension of the room from the globally aligned meshes of both systems. For marker-based system, the length of the laboratory room was found to be 6616.90 mm and width was 6513.22 mm. This is very

close from the actual length 6600 mm and width 6510 mm measured by tape. The error for width measurement was only 3.22 mm over such distance and for the length the error was only 16.9 mm. On the other hand, for the feature-based system the width for the laboratory room was found to be 6491.29 mm and length was 6512.16 mm. We can see that the readings are much lower than the actual measurements. The error for features based system for the width is 18.71 mm and length are 87.84 mm. The higher rate of error can be explained because, during global registration, feature based system tries to adjust each frame to reduce the global errors. While doing so, it may overcompensate the rotation. Even slight rotational change cause large error when measuring two point from long distance. Comparing the error of both of this system, we can conclude that, the marker-based system is much accurate and reliable and thus suitable for applications where accuracy is very important.

Chapter 6

Conclusion

In this thesis, we propose a method for an online marker-based 3D reconstruction system which can aide users in 3D reconstruction process interactively. To enable the user to freely move around we have developed a standalone platform. Using this platform, the user has no bound to be within certain area. The platform includes all the things needed to perform online 3D reconstruction. We have also conducted experiment to find the best camera rotation which is helpful to get accurate measurement. Upon our findings, we have designed a Graphical User Interface to aide the user to know which camera rotation they are taking the pose. So that the journey towards better accuracy begins at the time of image acquisition. We also help the user to know the current state of markers in the image in real time. We have tested our system in two different environments. One is a laboratory room and the other one residential room. Both are different in terms of texture and shape present in the room. Also, the size of these two environments are also different. To compare our marker-based system, we also developed a marker less feature-based system and test that system also in these two environments in the same conditions. Our experiments show that our marker-based system performs at a higher accuracy in terms of alignment. Also, in terms of accuracy of measurement, our system performs far better than the marker-less feature-based system. We have also shown that our marker-based system not only outperforms feature-based system in terms of accuracy but also in terms of number of poses required for 3D reconstruction. Not to mention, it achieves all of this in real-time while being deployed in a very low-priced computer having low computation power.

6.1 Limitations

While we have tried our best to reduce the dependency on any library and make our system available for any RGB-D camera, we must be dependent on the device manufacturer API for taking images. Although, third party libraries like OpenNI, Freenect, etc. usually develops drivers for RGB-D cameras available in the market, we have not yet heard of any news for Azure Kinect. Apart from the acquisition part, we only depend on OpenCV and PCL which are open source and universal.

6.2 Future Works

Currently our system depends on the user to acquire images which helps with the accuracy. Future works may involve in making this process autonomous. That means the system may automatically integrate new scene if did not see that part of the scene before. Another augmentation on the interactive part of this system could be suggestion of incorporating some part of the scene which could result in taking a smaller number of poses to fully reconstruct the scene. Moreover, volumetric integration of point cloud can also be thought of as an improvement, although implementing that in low configure PC would be hard. In addition to 3D reconstruction, other features such as real time scene segmentation and 3D object detection can also be added.

References

Atcheson, B., Heide, F., and Heidrich, W. (2010). Caltag: High precision fiducial markers for camera calibration. *VMV*, 10:41–48.

Beraldin, J.-A., Cournoyer, L., Rioux, M., Blais, F., El-Hakim, S. F., and Godin, G. (1997). Object model creation from multiple range images: acquisition, calibration, model building and verification. *Proceedings. International Conference on Recent Advances in 3-D Digital Imaging and Modeling (Cat. No. 97TB100134)*, pages 326–333.

Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. *Sensor fusion IV: control paradigms and data structures*, 1611:586–606.

Bobick, A. F. and Intille, S. S. (1999). Large occlusion stereo. *International Journal of Computer Vision*, 33(3):181–200.

Chen, Y. and Medioni, G. G. (1992). Object modeling by registration of multiple range images. *Image Vis. Comput.*, 10(3):145–155.

Coughlan, J. M. and Yuille, A. L. (1999). Manhattan world: Compass direction from a single image by bayesian inference. *Proceedings of the seventh IEEE international conference on computer vision*, 2:941–947.

Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., and Theobalt, C. (2017). Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4):1.

Dal Mutto, C., Zanuttigh, P., and Cortelazzo, G. M. (2012). *Time-of-flight cameras and Microsoft KinectTM*. Springer Science & Business Media.

De Bonet, J. S. and Viola, P. (1999). Poxels: Probabilistic voxelized volume reconstruction. *Proceedings of International Conference on Computer Vision (ICCV)*, pages 418–425.

DeGol, J., Bretl, T., and Hoiem, D. (2018). Improved structure from motion using fiducial marker matching. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 273–288.

Des Bouvrie, S. (2011). Improving rgbd indoor mapping with imu data.

Faugeras, O. and Keriven, R. (1997). Level set methods and the stereo problem. *International Conference on Scale-Space Theories in Computer Vision*, pages 272–283.

Fiala, M. (2005). Artag, a fiducial marker system using digital techniques. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2:590–596.

Hana, X.-F., Jin, J. S., Xie, J., Wang, M.-J., and Jiang, W. (2018). A comprehensive review of 3d point cloud descriptors. *arXiv preprint arXiv:1802.02297*.

Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2012). Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663.

Holzer, S., Rusu, R. B., Dixon, M., Gedikli, S., and Navab, N. (2012). Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2684–2689.

Ishikawa, H. and Geiger, D. (1998). Occlusions, discontinuities, and epipolar lines in stereo. *European conference on computer vision*, pages 232–248.

Jatavallabhula, K. M., Iyer, G., and Paull, L. (2019). gradslam: Dense slam meets automatic differentiation. *arXiv preprint arXiv:1910.10672*.

Jin, H., Soatto, S., and Yezzi, A. J. (2003). Multi-view stereo beyond lambert. *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, 1:I–I.

Khoshelham, K. (2011). Accuracy analysis of kinect depth data. *ISPRS workshop laser scanning*, 38(1).

Kim, S. and Woo, W. (2005). Projection-based registration using a multi-view camera for indoor scene reconstruction. *Fifth International Conference on 3-D Digital Imaging and Modeling*

(3DIM'05), pages 484–491.

Klopschitz, M. and Schmalstieg, D. (2007). Automatic reconstruction of wide-area fiducial marker models. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 71–74.

Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., et al. (2000). The digital michelangelo project: 3d scanning of large statues. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 131–144.

Lim, H. and Lee, Y. S. (2009). Real-time single camera slam using fiducial markers. *2009 ICCAS-SICE*, pages 177–182.

Liu, C., Schwing, A. G., Kundu, K., Urtasun, R., and Fidler, S. (2015). Rent3d: Floor-plan priors for monocular layout estimation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3413–3421.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the seventh IEEE international conference on computer vision*, 2:1150–1157.

Madeira, T., Oliveira, M., and Dias, P. (2020). Enhancement of rgb-d image alignment using fiducial markers. *Sensors*, 20(5):1497.

Masuda, T. and Yokoya, N. (1995). A robust method for registration and segmentation of multiple range images. *Computer vision and image understanding*, 61(3):295–307.

Melbouci, K., Collette, S. N., Gay-Bellile, V., Ait-Aider, O., and Dhome, M. (2016). Model based rgbd slam. *2016 IEEE International Conference on Image Processing (ICIP)*, pages 2618–2622.

Microsoft (2019). Azure kinect dk hardware specifications.

Munoz-Salinas, R., Marin-Jimenez, M. J., and Medina-Carnicer, R. (2019). Spm-slam: Simultaneous localization and mapping with squared planar markers. *Pattern Recognition*, 86:156–171.

Naik, N., Zhao, S., Velten, A., Raskar, R., and Bala, K. (2011). Single view reflectance capture using multiplexed scattering and time-of-flight imaging. *Proceedings of the 2011 SIGGRAPH Asia Conference*, pages 1–10.

Neunert, M., Bloesch, M., and Buchli, J. (2016). An open source, fiducial based, visual-inertial motion capture system. *2016 19th International Conference on Information Fusion (FUSION)*, pages 1523–1530.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 23(1):127–136.

Nießner, M., Zollhöfer, M., Izadi, S., and Stamminger, M. (2013). Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):1–11.

Olson, E. (2011). Apriltag: A robust and flexible visual fiducial system. *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407.

Rother, C. and Carlsson, S. (2001). Linear multi view reconstruction and camera recovery. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 1:42–50.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152.

Rusu (2019a). Fast point feature histograms (fpfh) descriptors.

Rusu (2019b). Point feature histograms (pfh) descriptors.

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. *2009 IEEE international conference on robotics and automation*, pages 3212–3217.

Seitz, S. M. and Dyer, C. R. (1999). Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173.

Shabalina, K., Sagitov, A., Svinin, M., and Magid, E. (2018). Comparing fiducial markers performance for a task of a humanoid robot self-calibration of manipulators: A pilot experimental study. *International Conference on Interactive Collaborative Robotics*, pages 249–258.

Son, H., Kim, C., and Kim, C. (2015). 3d reconstruction of as-built industrial instrumentation models from laser-scan data and a 3d cad database based on prior knowledge. *Automation in Construction*, 49:193–200.

Sun, J., Li, Y., Kang, S. B., and Shum, H.-Y. (2005). Symmetric stereo matching for occlusion handling. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*

(CVPR'05), 2:399–406.

Tamaazousti, M., Gay-Bellile, V., Collette, S. N., Bourgeois, S., and Dhome, M. (2011). Non-linear refinement of structure from motion reconstruction by taking advantage of a partial knowledge of the environment. *CVPR 2011*, pages 3073–3080.

Tombari, F., Salti, S., and Di Stefano, L. (2010). Unique signatures of histograms for local surface description. *European conference on computer vision*, pages 356–369.

Tombari, F., Salti, S., and Di Stefano, L. (2011). A combined texture-shape descriptor for enhanced 3d feature matching. *2011 18th IEEE international conference on image processing*, pages 809–812.

Turk, G. and Levoy, M. (1994). Zippered polygon meshes from range images. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318.

Wang, J. and Olson, E. (2016). Apriltag 2: Efficient and robust fiducial detection. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198.

Wang, K., Zhang, G., and Bao, H. (2014). Robust 3d reconstruction with an rgb-d camera. *IEEE Transactions on Image Processing*, 23(11):4893–4906.

Wang, S., Fidler, S., and Urtasun, R. (2015). Lost shopping! monocular localization in large indoor spaces. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2695–2703.

Whelan, T., Kaess, M., Fallon, M., Johannsson, H., Leonard, J., and McDonald, J. (2012). Kintinuous: Spatially extended kinectfusion.

Whelan, T., Kaess, M., Leonard, J. J., and McDonald, J. (2013). Deformation-based loop closure for large scale dense rgb-d slam. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 548–555.