AMERICAN UNIVERSITY OF BEIRUT

RGB-D CORRECTION AND COMPLETION AND ITS APPLICATION TO SLAM IN FEATURE-POOR PLANAR ENVIRONMENTS

by

JEAN WADIH LAHOUD

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Mechanical Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
April 2014

AMERICAN UNIVERSITY OF BEIRUT

RGB-D CORRECTION AND COMPLETION AND ITS APPLICATION TO SLAM IN FEATURE-POOR PLANAR ENVIRONMENTS

by

JEAN LAHOUD

Approved by:

[illegible]

Daniel Asmar, Assistant Professor | Advisor
Mechanical Engineering

[illegible]

Elie Shammas, Assistant Professor | Committee Member
Mechanical Engineering

[illegible]

Najib Metni, Assistant Professor | Committee Member
Mechanical Engineering
Notre Dame University, Zouk Mosbeh, Lebanon

Date of thesis defense: April 17, 2014

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to all the individuals without whom this thesis would not have been completed, and to whom I am greatly indebted.

I would like to thank my advisor, Dr Daniel Asmar, for his continuous support, motivation, and guidance during my work on this thesis. Without your help this thesis would not have been possible. My warmest thanks also goes to Dr Bernard Ghanem and Dr Ali Thabet for offering me the opportunity to visit King Abdullah University of Science and Technology and work in their group to advance my research work. A special thank you also to committee members, Dr Najib Metni and Dr Elie Shammas, for their encouragement and suggestions during my work on this thesis.

I would also like to acknowledge my colleagues, friends, and staff members at AUB and KAUST who were helpful during the research work.

To my family I dedicate this work. To my parents, who are always a source of encouragement and inspiration to me throughout my life, a very special thank you for actively supporting me in my determination to find and realize my potentials. To my sisters, Sanaa and Rima, thank you for your support and motivation.

Last but not least, a very special thank you to my friend, Mireille, for supporting me all the way and assisting me to reach my goals.

AN ABSTRACT OF THE THESIS OF

Jean Wadih Lahoud for Master of Engineering
Major: Mechanical Engineering

Title: RGB-D Correction and Completion and its Application to SLAM in Feature-poor Planar Environments

This thesis focuses on using an RGB-D sensor (Microsoft Kinect) for localization and mapping in an indoor planar environment. Such environments need special treatment since commonly used algorithms such as feature tracking or Iterative Closest Point (ICP) fail due to lack of visual features and 3D variations within the range of the Kinect. The idea is to make use of the in-range depth data, the 2D appearance data provided by the Kinect, and prior knowledge on the environment (planar) to correct, complete, and extend the 3D information beyond the range of the Kinect sensor. The sensor noise is used to robustly and adaptively fit planes through data points and use a properly designed Markov Random Field (MRF) to label pixels that are consistent with the scene, both in 2D and 3D. As such, depth is inferred at pixels with unknown depth values. After depth correction and extension, an adaptive and robust SLAM method is presented. This novel method uses the new depth data to find the transformation that best registers two consecutive frames. Both feature-point and plane matches are used to improve registration. We evaluate our method on two datasets and show its advantages over other RGB-D SLAM methods.

CONTENTS

Page

ACKNOWLEDGEMENTS . . . . **v**

ABSTRACT . . . . **vi**

LIST OF ILLUSTRATIONS . . . . **ix**

LIST OF TABLES . . . . **xii**

Chapter

I. Introduction **1**

II. Background **5**

A. Camera Model 5

B. Camera Calibration 8

C. Structured Lighting 9

D. Kinect Sensor 12

E. Iterative Closest Point Algorithm (ICP) 14

F. Image Feature Detection and Matching 18

G. Simultaneous Localization and Mapping (SLAM) 19

III. Literature Review **21**

A. Depth Enhancement Techniques for RGB-D Sensors 21

B. Current RGB-D SLAM Methods 23

IV. 3D Aware Correction and Completion of Depth Maps in Peicewise Planar Scenes **25**

A. Depth Measurement Noise: 27

B. Pre-Processing 29

1. Adaptive Depth Smoothing: 30
2. Adaptive and Robust Plane Fitting: 31

C. Depth Completion as an MRF 34

1. Determining Background Pixels: 34
2. Discriminative Appearance Model: 35
3. Vanishing Lines: 36
4. MRF Formulation 36
5. Iterative Solution: 39

V. RGB-D SLAM in Feature-Poor Planar Environments . **41**

VI. Experimental Results **43**

A. Dataset Compilation 43

B. Quantitative Comparison 45

1. Result on NYU Dataset [37] 47

C. RGB-D SLAM 48

D. Depth Upsampling 51

VII. Conclusion and Future Work **53**

Bibliography **54**

ILLUSTRATIONS

Figure Page

1. Examples of areas in which regular RGB-D SLAM methods fail. 2
2. Pinhole Camera Model 6
3. Pinhole Camera Model Reference Frame 6
4. Perspective Projection 7
5. An example of pattern projection for 3D reconstruction using structured lighting 9
6. Laser triangulation 9
7. The triangulation principle 11
8. The Kinect Sensor 12
9. Kinect Pattern 13
10. Kinect pattern projection on close objects 13
11. Kinect Color and Depth Data 14
12. Point-to-plane distance - ICP 16
13. Robot in corridor with corners 17
14. ICP solution for robot motion estimation in corridor with corners. 17
15. Robot in corridor without corners 18
16. ICP solution for robot motion estimation in corridor without corners. . . . 18

17. SIFT feature matching between two hallway images. 19

18. Kinect reliable depth data 20

19. SIFT matches for pixels with known depth 20

20. Overview of the proposed method 26

21. Sample depth frame from the Kinect 27

22. Histogram of the percentage of pixels discarded in a typical office/lab environment 27

23. Plot of the standard deviation of the depth measurement as a function of distance to the sensor. 28

24. Mean and standard ddeviation of the error of the Kinect 30

25. Depth deception 30

26. The result of the adaptive depth smoothing 31

27. Another result of the adaptive depth smoothing 32

28. Initial labeling 35

29. Vanishing lines in a corridor 37

30. End-to-end process 40

31. Different views of the large 3D reconstructed dataset 44

32. Ground truth frames 45

33. Comparison of raw depth values and depth values generated by our method with respect to the ground truth 46

34. Percentage of depth values added to a single image 47

35. Comparison between the proposed method and a colorization method . . . 48

36. SLAM results in a hallway for dataset 1 49

37. SLAM results in a hallway for the whole loop of dataset 1 50

38. SLAM results in a hallway for dataset 1 50

39. SLAM results in a hallway for dataset 2 50

40. SLAM error 51

41. Depth upsampling results 52

TABLES

Table Page

1. Average number of inlier feature matches 49

CHAPTER I

INTRODUCTION

Using RGB-D cameras such as the Microsoft Kinect to build dense 3D maps of indoor environments is of great importance in various applications such as robot navigation. This process is commonly referred to as RGB-D SLAM (Simultaneous Localization and Mapping) and has been studied in various fields, including robotics and computer vision. Using these sensors is appealing as they are relatively cheap, accessible, and well supported by manufacturers and developers. The main advantage of these devices is in their ability to capture 3D data directly without needing to infer 3D shape from 2D appearance, which in general is not trivial.

RGB-D SLAM algorithms usually start by extracting visual features from a frame and then match them to features in other frames. These matched points along with their depth information are then used to track the motion of the sensor. These algorithms tend to fail in environments that lack enough visual features or when the detected features lack depth information due to sensor limitations. Such limitations stem from range limits and inaccuracies due to projections on IR-absorbing or reflective surfaces. Moreover, SLAM algorithms that rely solely on depth information, by performing ICP on the data, also fail in environments that lack 3D geometric variations. This mainly happens in hallways and corridors similar to the ones shown in Fig. 1. In such cases, alternative techniques should be used to perform SLAM.

In this thesis, a novel RGB-D SLAM method is presented for a typical indoor environment (piecewise planar), where known depth data along with 2D appearance information are used to infer additional depth information and plane segments. This thesis studies how reliable depth values can be used to *correct* unreliable depth values, when a prior model is assumed on the 3D scene, and how to *complete* (or extend) the depth values beyond the raw measurements of the sensor (*i.e.*, infer depth at pixels with unknown depth values). Piecewise planar environments are considered, since many indoor scenes with man-made objects can be modeled as such. The proposed framework uses the RGB-D sensor's noise profile to adaptively and robustly fit plane segments (*e.g.* floor and ceiling) and iteratively complete the depth map, when possible. Depth completion is formulated as a discrete labeling problem (Markov Random Field) with hard constraints and solved efficiently using graph cuts. To regularize this problem, 3D and appearance cues are exploited for encouraging pixels to take on depth values that will be compatible in 3D to the planar assumption. The new depth information can then be used for better motion estimation. Since benchmark RGB-D SLAM methods treat the whole depth data equally and do not exploit all possible image features, a modified method is proposed for a more robust motion estimation. The proposed approach comprises a two-step RGB-D SLAM method. It first exploits all features with depth information to find a first estimate of motion. It then uses the feature matches to find matching planes and perform an appearance and depth consistent motion estimation.

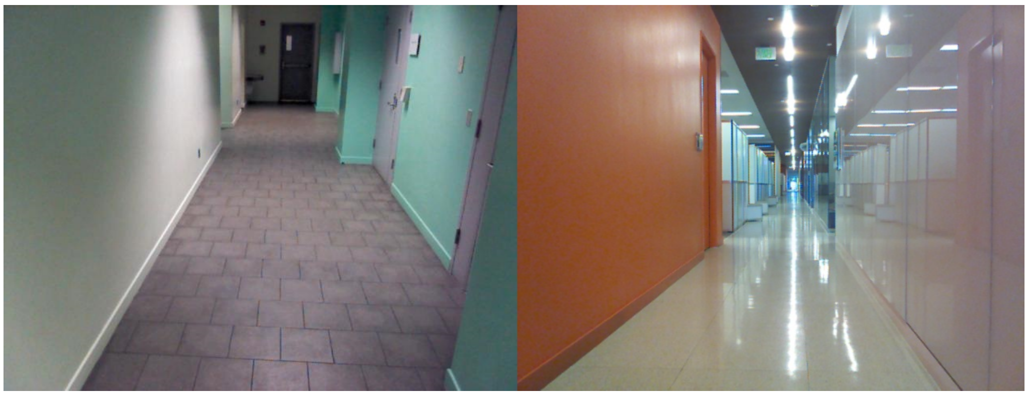
Figure 1: Examples of areas in which regular RGB-D SLAM methods fail.

Contributions: They are four fold.

- Unlike other depth enhancement methods, this thesis addresses the problem of correction and completion of unreliable and unknown depth values in a single RGB-D image pair. To the best of my knowledge, this is the first work that makes use of local and global priors on the overall 3D scene to enable 3D aware depth prediction and correction. Here, the global prior on the scene is that it is a piecewise planar environment.

- Instead of discarding Kinect unreliable depth information, its noisy (probabilistic) structure is used to perform adaptive depth smoothing and adaptive robust plane fitting. The depth correction/completion process is modeled as a discrete MRF (labeling problem) that can be efficiently solved using iterative interactive graph cuts. The unary and binary terms of the MRF go beyond traditional definitions to stress appearance and 3D cues that encourage a 3D structure compatible with the planar assumption.
- To evaluate the proposed approach and validate the importance of depth correction and completion in piecewise planar environments, a challenging, large-scale ground truth dataset is compiled. Also, the merit of the solution is illustrated in two real-world applications: RGB-D SLAM and depth upsampling.
- Instead of having all depth points contribute equally for the final transformation of the RGB-D SLAM technique, weights are introduced to the added depth points according to their MRF scores. MRF scores signify the accuracy of the depth points added, among which the more accurate is chosen to have a higher contribution for the final motion estimation.

Thesis Outline: Chapter II presents a background review on the concepts needed for this thesis. The camera model is first introduced as well as camera calibration. A review on the sensor used in this thesis (Kinect) is also presented with the concept of structured lighting that it utilizes. Methods for motion estimation, such as the ICP algorithm and feature-based SLAM, are then introduced along with their limitations.

Chapter III is a literature review survey on work that is related to the subject considered. This chapter revises depth enhancement techniques for RGB-D sensors as well as current RGB-D SLAM methods.

Chapter IV is focused on the proposed correction and completion method of depth maps in piecewise planar scenes. The first section deals with the sensor noise and the use of unreliable depth data. The pre-processing steps are then described, which include the adaptive depth smoothing and the adapative and robust plane fitting. The depth completion method is then portrayed as an MRF problem that is aware of the 3D scene.

Chapter V presents the RGB-D SLAM method that can be used in feature-poor planar environments. This method uses the enhanced depth frames from chapter IV and introduces modification to regular RGB-D SLAM methods to improve the results.

Chapter VI includes the experimental results for the proposed method. An RGB-D dataset is compiled to test the proposed technique, and results upon applying the technique on a known dataset are also presented. RGB-D SLAM and depth upsampling results are also shown.

Finally, chapter VII concludes the thesis. Future work is also suggested , which include ideas that can be implemented in the future.

CHAPTER II

BACKGROUND

This chapter provides information on the basic principles that were utilized in this thesis and some of the challenges faced for solving the problem addressed.

A. Camera Model

Images are 2D projections of real world scene. A mathematical model is necessary in order to establish a mathematical relationship between the coordinates of a 3D point and its projection onto the image plane. Camera models include:

- Pinhole camera model
- Orthographic projection
- Scaled orthographic projection
- Paraperspective projection
- Perspective projection

We here restrict our study to the pinhole camera model and the perspective projection which were used in this thesis. The reader may refer to [19] for a detailed study on camera models.

Pinhole Camera Model This model describes the projection of 3D point onto an ideal pinhole camera, which does not include any lens. Fig. 2 shows how points from a 3D object are projected onto the image plane of a pinhole camera.

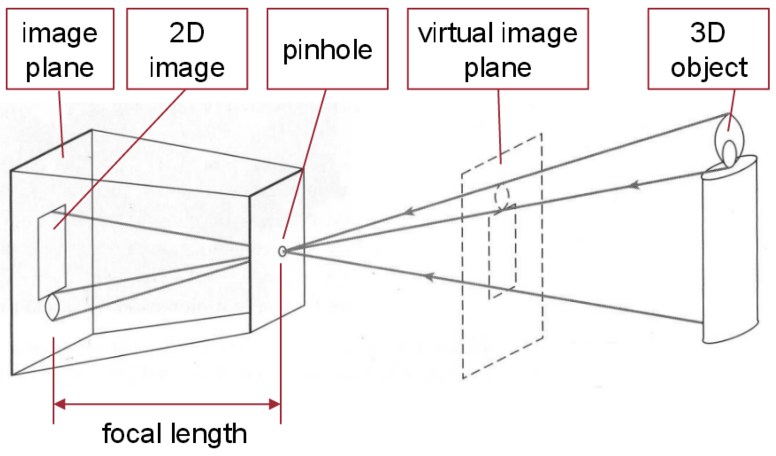

Figure 2: Pinhole Camera Model

Given a 3D point $\mathbf{P} = (X,Y,Z)^T$ which projects to a 2D point $\mathbf{p} = (x,y)^T$, we define an image frame and a world frame as shown in Fig. 3.

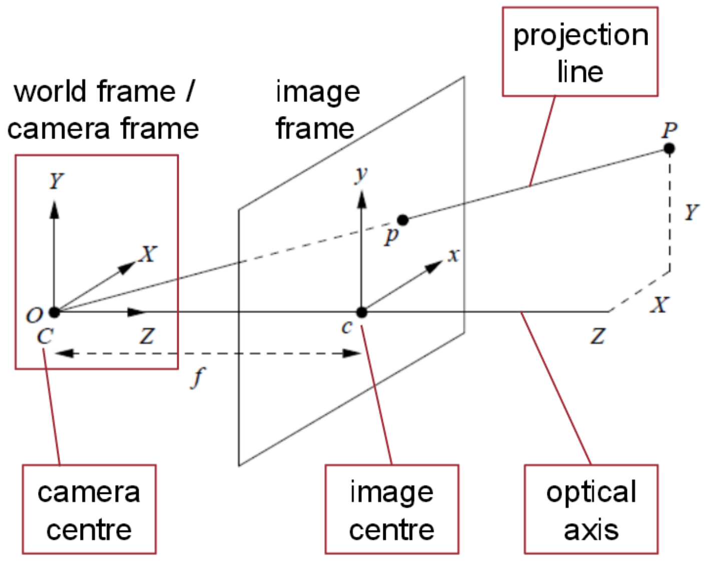

Figure 3: Pinhole Camera Model Reference Frame

By Thales rule, we have

$$\frac{X}{Z} = \frac{x}{f} \qquad and \qquad \frac{Y}{Z} = \frac{y}{f}$$

or

$$x = f\frac{X}{Z} \qquad and \qquad y = f\frac{Y}{Z}$$

Perspective Projection This is a more complex representation of a camera model which better describes a real camera. Since camera sensor's pixels are not exactly square, focal lengths along *x* and *y* can be scaled, yielding

$$x = f_x \frac{X}{Z} \qquad \textit{and} \qquad y = f_y \frac{Y}{Z}$$

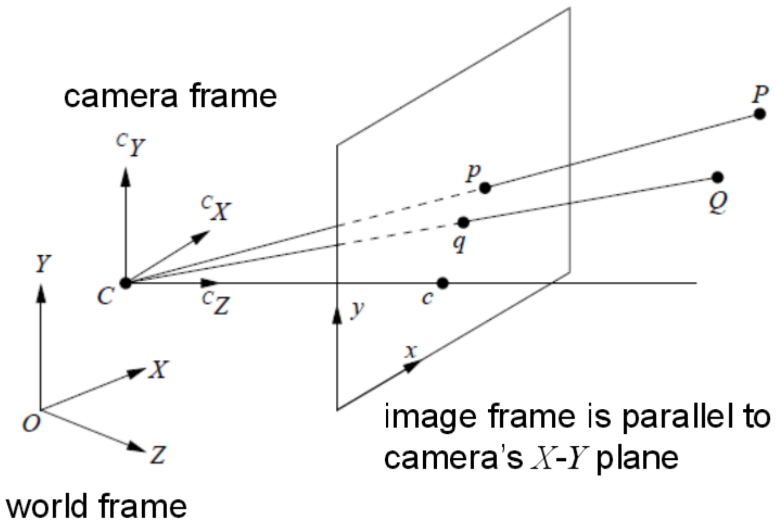

Figure 4: Perspective Projection

The image center or principal point *c* may not be at the origin, therefore

$$x = f_x \frac{X}{Z} + c_x \qquad \textit{and} \qquad y = f_y \frac{Y}{Z} + c_y$$

where c_x and c_y are the location of *c* in the image plane. Moreover, cameras may have frames that are not exactly rectangular. In this case,

$$x = f_x \frac{X}{Z} - f_x \cot\theta \frac{Y}{z} + c_x \qquad \textit{and} \qquad y = \frac{f_y}{\sin\theta} \frac{Y}{Z} + c_y$$

where θ is the skew angle between the x-axis and the y-axis. This from can be written in matrix from as

$$\tilde{\mathbf{x}} = \frac{1}{Z} \mathbf{KX}$$

where

$$\mathbf{K} = \begin{pmatrix} f_x & -f_x \cot\theta & c_x \\ 0 & \dfrac{f_y}{\sin\theta} & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

and $\tilde{\mathbf{x}} = [x, y, 1]^T$ are the homogeneous coordinates. $\mathbf{K}$ is the intrinsic parameter matrix which can be written in a simpler form using the skew parameter s

$$\mathbf{K} = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Since the camera frame may not be aligned with the world frame, a rigid transformation should be defined that relates the coordinates of a 3D point in the camera frame to its coordinates in the world frame. The transformation is defined as follows:

$${}^{C}\mathbf{X} = {}^{C}_{W}\mathbf{R}\,{}^{W}\mathbf{X} + {}^{C}_{W}\mathbf{T}$$

The rotation and translation matrices represent the extrinsic camera parameters.

B. Camera Calibration

Camera calibration refers to the process of finding the intrinsic and extrinsic parameters of a camera. During calibration, a set of points with known positions (such as a checkerboard edges) is imaged in a fixed world coordinate system. The calibration process is an optimization problem that minimizes the difference between the observed and the theoretical location of the points as predicted by the perspective projection equations. Calibration techniques can be found in [22, 40, 43]. A Matlab toolbox for camera calibration can be found in [7].

C. Structured Lighting

The general principle of structured light is to project a known pattern onto a scene and infer depth from the deformation of that pattern. Structured lighting is closely related to stereo vision, as both are based on the fact that if two cameras observe the same scene point then its position can be retrieved by intersecting the rays corresponding to the projection on each image. This process is called triangulation. Passive techniques, such as stereo vision, face problems in matching points between two images for them to recover the 3D information. For this reason, active stereo techniques have been introduced, where one of the cameras is replaced by a calibrated and well defined light source that mark the scene with some known pattern [38].

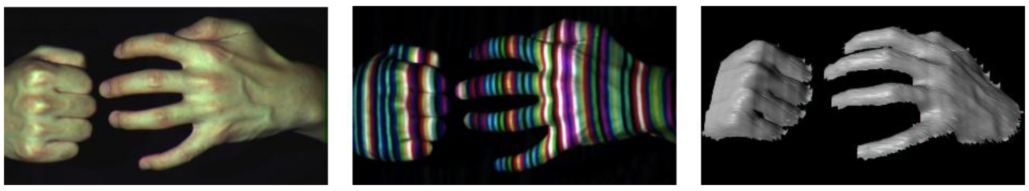

Figure 5: An example of pattern projection for 3D reconstruction using structured lighting

One of the early structured lighting techniques used a laser to direct a beam which is known to lie within a plane. The beam's position in the image is then detected. For every beam point, a ray emerging from the camera is intersected with the plane of the laser, and the intersection represents the points position in 3D. The process is shown in Fig. 6.

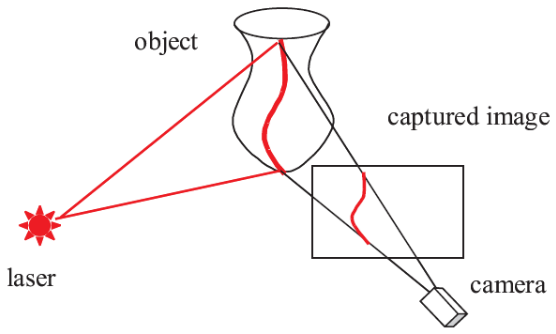

Figure 6: Laser triangulation

Early techniques which used a single beam resulted in only sparse reconstruction. In order to obtain dense reconstruction, multiple techniques have been introduced. One technique included moving the object of the light source in order to get a 3D reconstruction of the whole body. A major drawback for this technique is its inability to recover the shape of moving objects. Another technique projects multiple stripes onto the scene in order to improve speed and also recover moving objects. This requires coding of the stripes for the correspondence problem to be solved without ambiguity. Many patterns have been proposed for structured light. An implementation of a set of representative techniques and some comparative results can be found in [35]. All techniques start by the camera and projection calibration, then establish the correspondences between the image and the projected stripes, then reconstruct the 3D coordinates of the points in the scene using triangulation. The difference between all the techniques is in the way each solves the correspondence problem.

3D Reconstruction using the principle of triangulation Consider a 3D imaging system containing a camera sensor and a projector having a fixed relative position. Two different sets of parameters, one for the camera and the other for the projector, have to be computed by calibration. Projector calibration can be done using the Bouguet toolbox [7]. Note that 3D reconstruction from uncalibrated images is also possible but the problem is more challenging. Refer to [18] for a technique that uses uncalibrated images for reconstruction. Moreover, correspondences between projected points and points viewed by the camera should be established. Once the calibration parameters and the correspondences are known, one can use the principle of triangulation to reconstruct the 3D scene. This principle is outlined next.

Consider a point P_1 being projected as shown in Fig. 7. The projection of that point into the 3D space is then being projected onto the camera at point P_2. Since point correspondences have been already established, the coordinates of point P_2 are known. The 3D coordinates of the object point where the projecting point has been reflected on

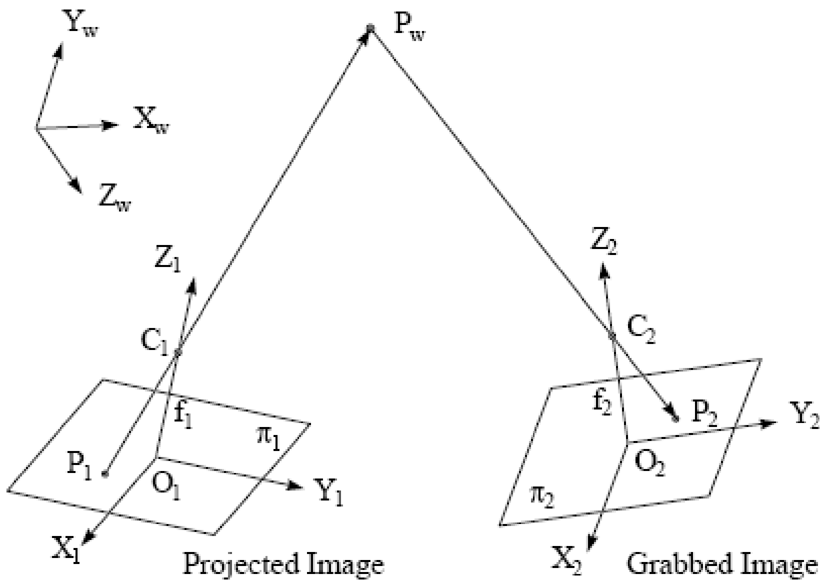

Figure 7: The triangulation principle

the image sensor P_w is to be computed. We limit our study to linear systems *i.e*, the calibration matrices do not include the lens distortion term. For the projector, we have

$$w_1 \begin{pmatrix} x_{P_1} \\ y_{P_1} \\ 1 \end{pmatrix} = \mathbf{K_1} \begin{pmatrix} x_{P_w} \\ y_{P_w} \\ z_{P_w} \\ 1 \end{pmatrix}$$

and for the camera, we have

$$w_2 \begin{pmatrix} x_{P_2} \\ y_{P_2} \\ 1 \end{pmatrix} = \mathbf{K_2} \begin{pmatrix} x_{P_w} \\ y_{P_w} \\ z_{P_w} \\ 1 \end{pmatrix}$$

where $\mathbf{K_1}$ and $\mathbf{K_2}$ are the calibration matrices containing the intrinsic and extrinsic parameters for the projector and camera respectively. The values of w_1 and w_2 are first replaced using the third entry of each set of equations. Re-arranging to solve for x_{P_w}, y_{P_w}, and z_{P_w}, we obtain a system of 4 equations with 3 unknowns that can be solved using least square minimization. Note the direct relation between the coordinates of the 3D point and those of the 2D points. Any error in the correspondences between the points in the pattern and those projected on the camera plane leads to error in the 3D reconstruction.

D. Kinect Sensor

RGB-D cameras, among which we choose the Microsoft Kinect, capture both color and depth information. The Kinect sensor, shown in Fig. 8, is based on the technology developed by PrimeSense [1] and uses structured lighting techniques to calculate depth. This is done by projecting a speckle pattern via an infrared projector and then capturing it via an infrared receiver. The pattern is shown in Fig. 9. In order to match the dots (speckles), window search and region growing are performed. Since the pattern is unique, corresponding points in the emitted and received pattern can be detected, and depth measurement can be done via triangulation. Moreover, the Kinect combines with structured light depth from focus. The depth from focus principle is based on the idea that objects become more blurry further away. This means that the size of the speckle dot increases with distance. The Kinect also uses an astigmatic lens with different focal length in the x and y direction so that projected circles become ellipses whose orientations depend on depth.

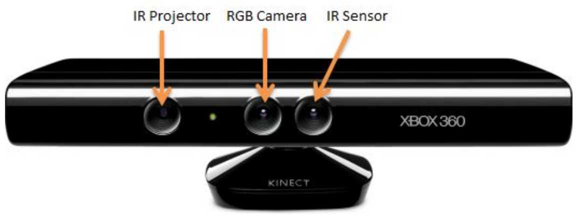

Figure 8: The Kinect Sensor

The advantages of using IR patterns include the ability to operate in low light conditions, on textureless objects, and on repeated scene textures. Nevertheless, IR

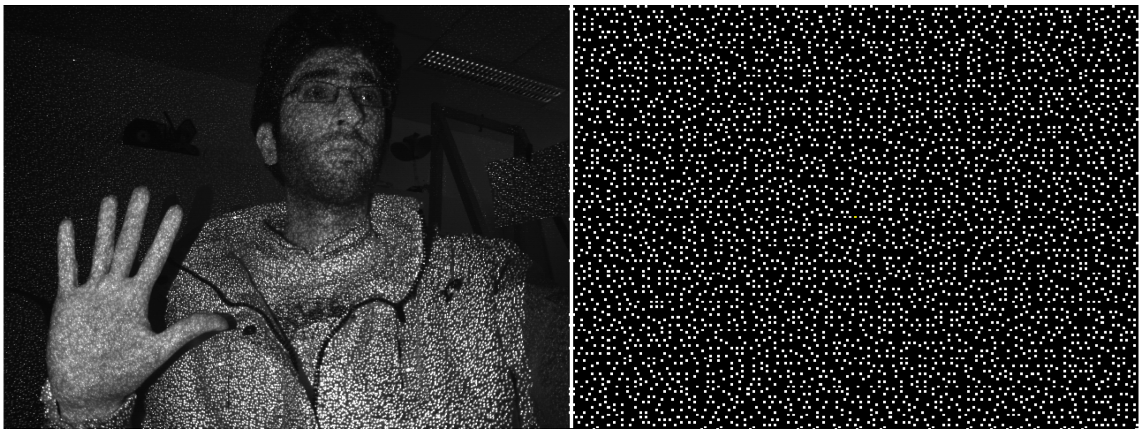

Figure 9: Kinect Pattern

projection does not work outside and uses more energy than stereo image capturing. Moreover, the range limitations arise due to the capturing sensor. Fig. 10 shows why close objects cannot be detected by the Kinect. As can be seen, light becomes too blurry at close distances which renders the speckle detection impossible.

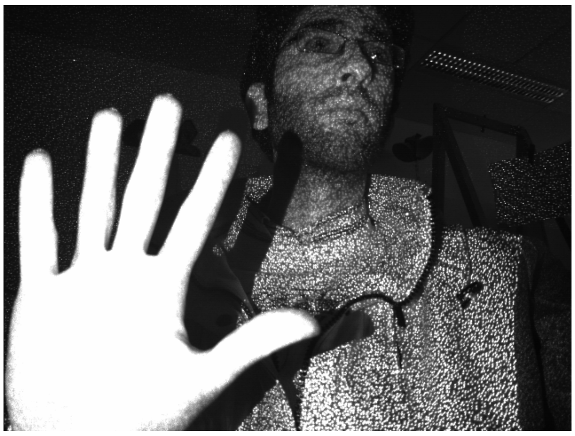

Figure 10: Kinect pattern projection on close objects

The depth frame captured by the Kinect is of size 640x480 points that correspond to a 640x480 image captured via a regular camera. The sensor thus provides a dense depth point cloud at 30 frames per second. The high resolution and frame rate come at a cost; the Kinect has a limited range (up to 4-5 meters), the depth estimate is not very accurate (more than 1% error), and the field of view is much less than that of specialized laser range finders (60° for the Kinect as opposed to 180° or even 360° for lasers). Fig. 11 shows RGB and depth frames obtained from the RGB-D camera. These represent the sensor data to be used throughout the whole project.

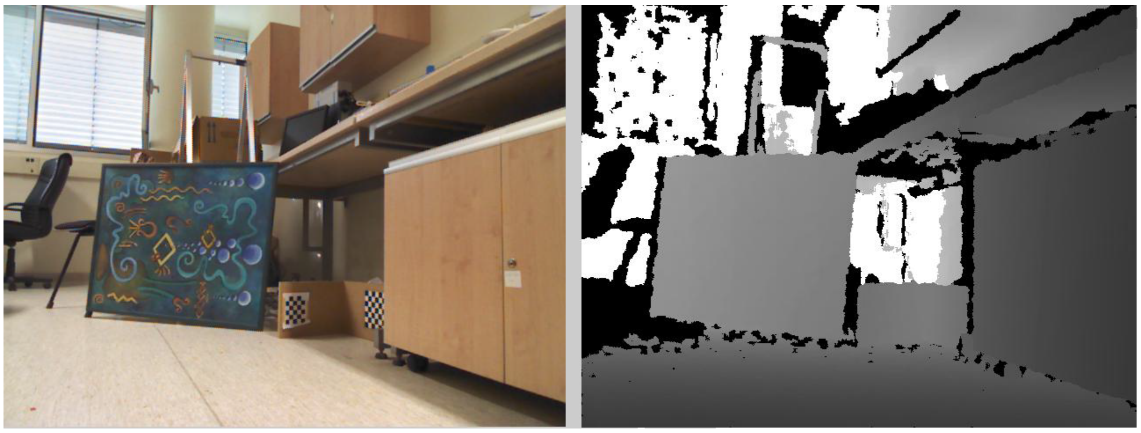

Figure 11: (Left) Color image captured by the Kinect. (Right) Depth image obtained. White and black pixels do not have depth information due to distance or occlusion.

E. Iterative Closest Point Algorithm (ICP)

The iterative Closest Point algorithm is used to register two point clouds by minimizing the difference between the two sets of points. Applications of ICP include reconstruction of surfaces from multiple scan and robot localization. The basic ICP algorithm is described in [6, 12, 42, 2]. The process consists of a source point that needs to be transformed to match a reference or target point cloud. The algorithm iteratively revises the transformation (translation and rotation) in order to minimize the distance between the two clouds. The basic outline of the ICP algorithm is as follows:

- For each point in the source point cloud, find the corresponding closest point in the other set
- Calculate the transformation that minimizes the error in distances between the matched points
- Apply the transformation found in the previous set to the source

- Iterate until stopping criteria is met (usually when error drops below a threshold, or the error variation between the iterations is minimal).

ICP have been proven to converge monotonically to a local minimum with respect to the mean squared error distance objective function [6]. Closest point association has two variants: point-to-point and point-to-plane. The latter usually have a better performance [34]. Variants of the ICP algorithm have differences in:

- Selecting the sample points from each of the point sets
- Matching the points in the source point cloud to the points in the reference point cloud (point-to-point or point-to-plane)
- Applying weights to the correspondences according to a given criteria
- Rejecting a certain number of the matched points (especially in cases that involve motion)
- Assigning an error metric to the current transform
- Minimizing the error metric with respect to the transformation

Fig. 12 shows the point to plane distance that should be minimized using the ICP. s_i represents the coordinates of the source point while d_j represents the coordinates of the destination point, and n_i is the unit normal to the destination surface. The following two sets are then given: $S = \{s_i\}_{i=1}^{N_s}$ and $D = \{d_j\}_{j=1}^{N_d}$. Note that every element s_i or d_i is a $3x1$ array that contains the coordinates of a certain point.

The objective function that we need to minimize is then

$$\sum_i \Big((M.s_i - d_i).n_i\Big)^2$$

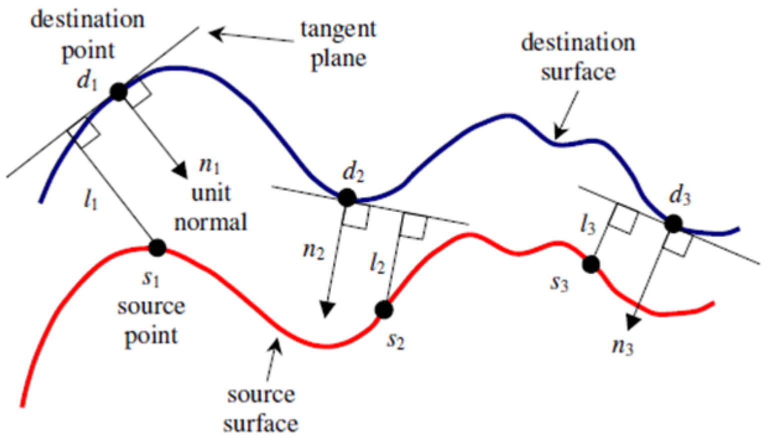

Figure 12: Point-to-plane distance - ICP

where M is the 4*x*4 transformation matrix representing the rotation and translation. The above function can also be written as a function of the rotation and translation as

$$\sum_i \left((R.s_i + t - d_i).n_i\right)^2$$

where *R* is a 3*x*3 rotation matrix representing a rotation about any axis in 3D space and *t* is a 3*x*1 translation matrix representing any translation in 3D. Note that the alternative to the above equation is the point to point distance where the normal component is omitted *i.e*, the distance is stated as

$$\sum_i (M.s_i - d_i)^2$$

ICP in Corridors This paragraph presents an example for the ICP algorithm usage as well as one of its limitations. Consider a robot moving in a corridor while scanning the area. For simplicity, the problem is considered in 2D (as in the case of 2D laser scanning), but can be extended to 3D with similar analysis. Two cases are considered: In the first case, the environment involves corners, while in the second, the scan does not contain any corner. The first case is represented in Fig. 13 where a robot moves between positions 1 and 2. In the first position, the robot scans the area for depth points, which are represented in red, and in the second, the depth points are represented in green. The

ICP process starts with the two point sets shown in Fig. 14 and matches the closest points, then iterate to minimize the distances between the two sets. Clearly, the minimum distance is that represented at the right of Fig. 14.

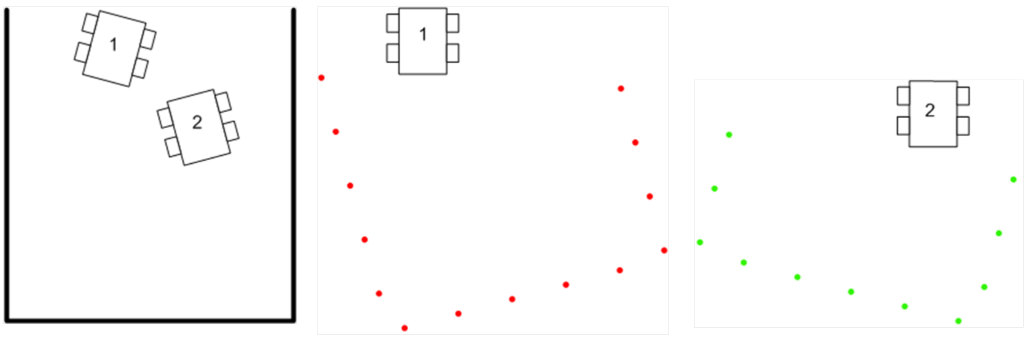

Figure 13: (left to right) The area scanned shown in black, depth captured by robot at position 1, and depth captured by robot at position 2.

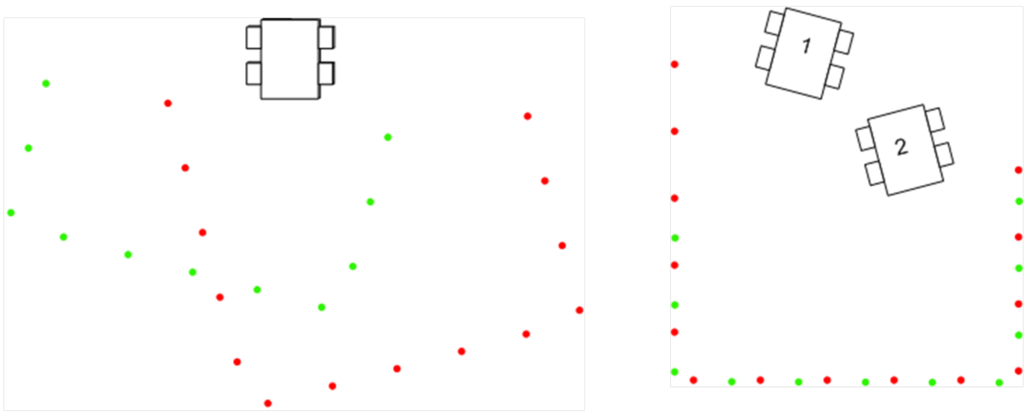

Figure 14: ICP solution for robot motion estimation in corridor with corners.

Now consider the case where the depth points do not include any corners as in Fig. 15. This occurs mainly due to the limitation in the sensor range, where the other end of the area lies beyond that range. The scans from the robot at the two positions are also shown in Fig. 15. Now, inputting the two sets into the ICP algorithm starts as in Fig. 16. Since the situation involves motion and the scanned areas are not identical, a portion of the point matches between the two sets has to be dropped. Nevertheless, the minimization problem has multiple solutions as shown in Fig. 16. The ICP process succeeded in finding the rotation of the robot but could not retrieve the translation.

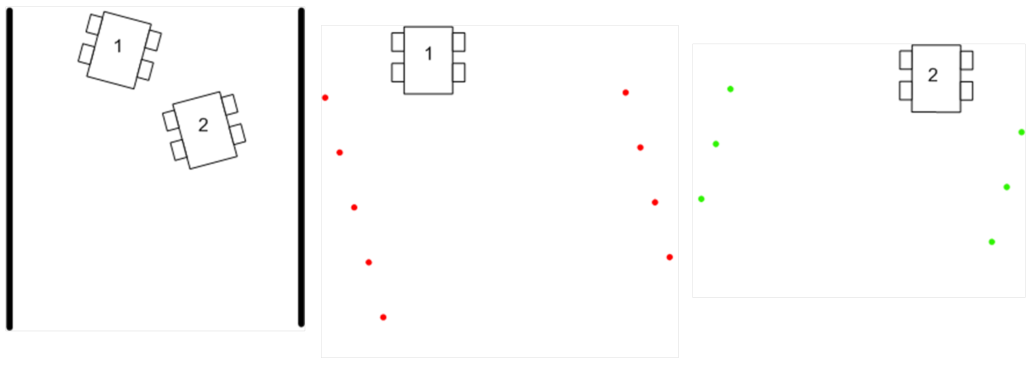

Figure 15: (left to right) The area scanned shown in black, depth captured by robot at position 1, and depth captured by robot at position 2.

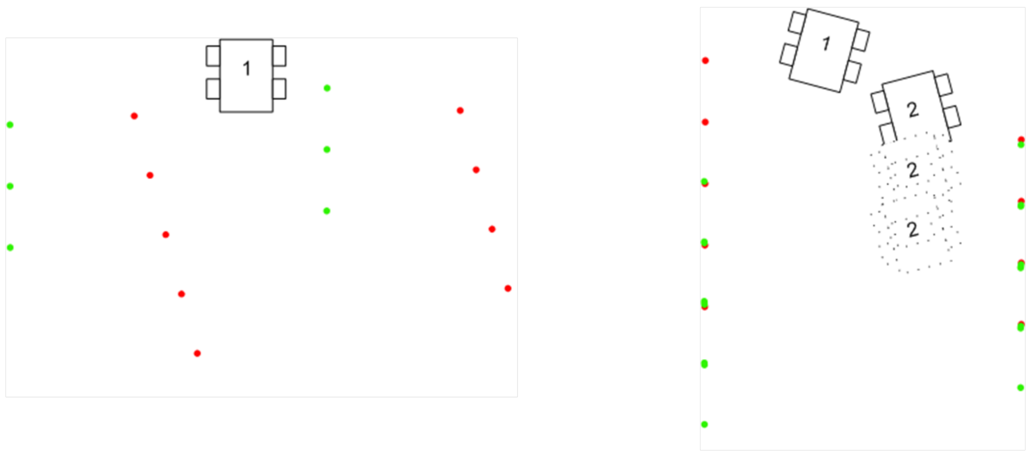

Figure 16: ICP solution for robot motion estimation in corridor without corners.

F. Image Feature Detection and Matching

A feature is an "interesting" part of an image for description. Image features can be isolated points, continuous curves, or connected regions. A good feature should be consistent over several images of the same scene, invariant to transformations, and insensitive to noise. Image features can be edges, corners/interest points, blobs/regions of interest, or ridges. Common applications of features are image alignment (*e.g.*, panoramic mosaics), object recognition, 3D reconstrution, and motion tracking. Image feature detection and matching is usually the starting point of many computer vision algorithms. Many techniques have been developed for feature description, such as Canny edge detector [11], Harris corners [21], Shi and Tomasi [36], SURF [4], and SIFT [31].

Fig. 17 shows an example of SIFT feature matching between two hallway images.

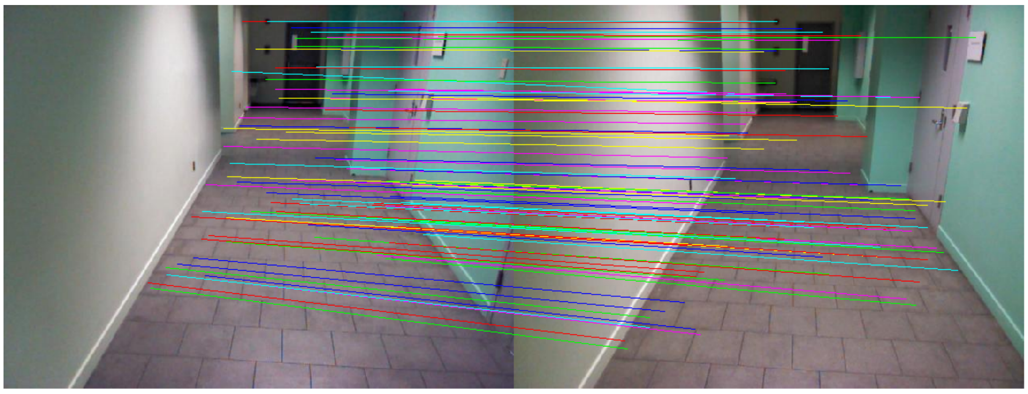
Figure 17: SIFT feature matching between two hallway images.

G. Simultaneous Localization and Mapping (SLAM)

The term SLAM is as stated an acronym for Simultaneous Localization and Mapping. It was originally introduced by Durrant-Whyte and Leonard [29] as SMAL but was later changed to SLAM. SLAM is a technique used by robots or autonomous vehicles to build up a map of the surrounding environment or to update a given map, while at the same time locating itself in the map. SLAM cam be sought as a chicken and egg problem: An accurate map is required for an accurate pose estimation whilst at the same time the motion estimation is needed to build the map. SLAM uses statistical techniques such as Kalman filters, particle filters (also known as the Monte Carlo methods) and scan matching of range data. These techniques provide an estimation of the posterior probability function for the both the robot pose and the map parameters. Apart from regular SLAM methods, RGB-D SLAM refers to the technique used for registering point clouds from RGB-D sensors such as the Kinect or stereo cameras. Registration of point clouds is equivalent to estimating the motion and building the map represented by these point clouds.

Common RGB-D SLAM methods start by feature detection and matching between frames. Feature matches whose depth data is known are then used for motion estimation. Failure of such algorithms happen when matches are insufficient or erroneous. For example, consider the hallway images in Fig. 17. Pixels whose depth data is known are shown in Fig. 18. Although one can easily detect that the robot moved in the original images, the motion is not detectable in these images, which makes motion estimation a tedious problem. Applying SIFT detection and matching to these images results in erroneous matches as seen in Fig. 19. Two problems are faced: the number of matches is low, and matches are incorrect due to the repeating tiles on the floor. Therefore, regular RGB-D SLAM techniques fail in such environments.

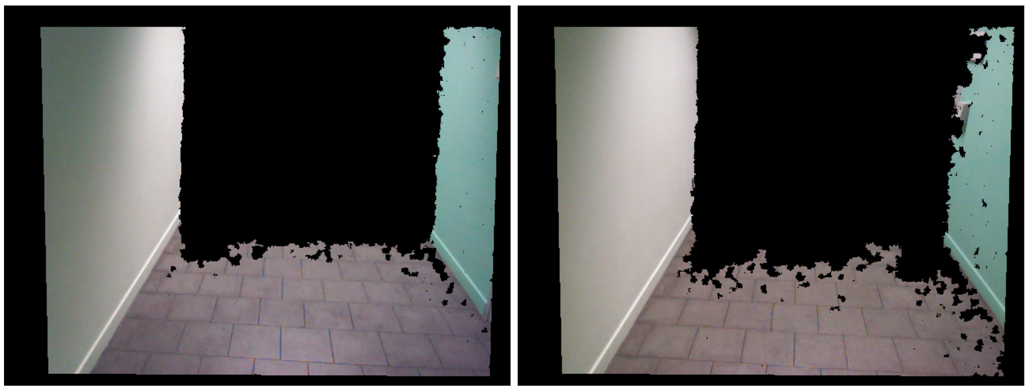
Figure 18: Kinect reliable depth data: Black pixels correspond to areas with unknown depth.

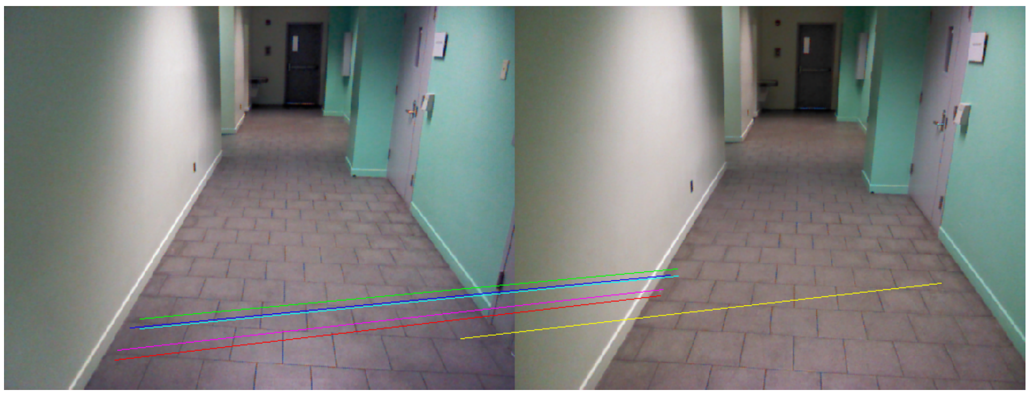
Figure 19: SIFT matches for pixels with known depth

A survey on related work is shown next.

CHAPTER III

LITERATURE REVIEW

The problem addressed in this thesis can be divided into two parts: depth enhancement for depth sensors such as the MS Kinect, and RGB-D SLAM. Related work usually tackles each problem independently.

A. Depth Enhancement Techniques for RGB-D Sensors

This work addresses the problem of correcting unreliable depth values and inferring unknown ones in RGB-D data of piecewise planar scenes. In data of this kind, large groups of contiguous pixels have either unreliable or unknown depth values. The most related work in the literature that address a similar problem (depth enhancement) can be categorized as: **(1)** hole-filling (depth inpainting) methods or **(2)** depth upsampling methods.

Methods of category **(1)** address the problem of inferring the depth of pixels that are not assigned depth values. Such pixels tend to be projections of parts of surfaces that are IR absorbing, reflective, or too close to the RGB-D sensor. These pixels are small in number and tend not to cluster in the same portion of a depth image. Most hole-filling methods interpolate (or propagate) unknown depths from depths in pixel neighborhoods. To this end, joint bilateral filtering has been extensively used to fill holes in depth images, especially due to its relatively small computational burden [27, 10]. Moreover, colorization techniques are also used to fill in unknown depth [30], as done in compiling the popular NYUv2 dataset [37]. In [14], the problem is formulated as a continuous Markov Random Field (MRF), where the latent variables are the depth values of all pixels, the unary (data) term is dependent on the known depth values, and the binary term encourages similar looking pixels in a local neighborhood to have similar depth values. In [13], a foreground depth model is assumed to be available and depth layers are inferred using a discrete MRF. Many other methods of this category exist (*e.g.*, [41]); however, they all suffer from the same drawback that makes them infeasible and inappropriate for the depth correction and completion problem in this thesis. Hole-filling (depth inpainting) methods assume that there is a strong correlation between depth discontinuities and image edges and that pixels with similar appearance have similar 3D geometry. In general, this assumption is a useful cue for interpolation but it does not always hold, especially in cases where large portions of the depth image need to be filled, which is the scope of this thesis.

Methods of category **(2)** address the problem of generating a high resolution depth image from a low-resolution depth image and (usually) a registered high resolution RGB image. The low-resolution depth image is usually assumed to be complete and comprising of reliable depth values. Since a plethora of such methods abound in the literature, we mention a representative few here. In fact, joint bilateral filtering and MRF labeling are common techniques employed by methods of this category [33, 27, 14]. Similar to the problem of hole-filling, local assumptions of depth smoothness (except at color discontinuity) are made to propagate depth values from the low resolution image to a local neighborhood of unknown values in the higher resolution image. These assumptions break down in the case of large contiguous holes, which makes upsampling methods unsuitable for the problem addressed in this thesis.

All previous methods apply local priors to depth values in RGB-D data, but they tend not to take into account the global 3D structure of the scene for unreliable depth correction and unknown depth completion. These methods do *not* ensure that the processed depth maps they produce lead to a 3D point cloud that has a compatible 3D structure (*i.e.*, its structure does not lead to any 3D contradictions or impossibilities). Assuming a piecewise planar scene allows us to regularize the completion and correction process globally as well as locally. This regularization disallows certain depth assignments that lead to a 3D structure that is not compatible or does not respect the planar assumption.

B. Current RGB-D SLAM Methods

The problem of RGB-D SLAM and building a 3D map has been tackled extensively in the literature. We here restrict our attention to systems that use RGB-D cameras [24, 32, 16, 25, 3]. We differentiate three types of techniques. Methods of the first type only use depth information to estimate motion while the color information is only used for visualization purposes. In the second type of methods, only color information (RGB without the depth) is used for motion prediction. This is usually referred to the in the literature as *monocular SLAM*. The depth information is then used to build a dense 3D reconstruction of the environment. The third type combines both depth and color information to predict motion and build the map.

When depth data is used for frame alignment, the ICP algorithm [6] and its many variants are commonly used. Variants of the ICP algorithm that minimize point to plane distances instead of point to point distances have shown an improvement in frame registration. This version of ICP is used to register two RGB-D frames in [32], where the ICP algorithm is used to track the sensor motion and then accumulate observations in the environment model. Localization is refined after several observations to ensure consistency. The limitation of using the ICP algorithm for tracking lies in the need for 3D geometric gradients, which are necessary for successful registration. Therefore, earlier RGB-D SLAM methods work well in structured rooms with corners, edges, and other 3D features but fail when scanning a straight wall even if visual features exist.

Tracking and mapping using RGB-D cameras can be split into two separate problems. Tracking can be done using the RGB data only, similar to using a monocular camera, while mapping is performed using the depth data. Motion detection between two RGB frames relies on tracking features from frame to frame, so the process is known to be feature-based. Popularly used features include SIFT [31], SURF [4], and FAST descriptors based on random trees [9]. When features are tracked, projective geometry is used to define spatial information for the features. Alignment is then done by minimizing a matching cost between matched points. This is done after removing outliers using RANSAC [17]. Such methods fail in environments that lack a sufficient number of visual features.

Using either color (features) or depth (appearance) from RGB-D cameras forgoes valuable data that can be useful for motion estimation. Performing monocular SLAM while using RGB-D cameras does not take advantage of the depth data; monocular SLAM re-estimates depth, up to a scale, for a certain number of feature points and registers these points only. Thus, it does not provide a motion estimation that best registers all the depth points of two frames. Therefore, recent methods have proposed to jointly optimize visual features and ICP. Henry et al. [24] proposed a full 3D mapping system that combines visual and depth information into a single joint optimization problem. This method minimizes an error function containing both the distance between tracked visual features and point-to-plane error of the ICP algorithm. Furthermore, other SLAM approaches [15, 3, 25] rely on feature matching and 3D cues, but do not attempt to correct/extend depth in order to incorporate more feature points into the motion estimation process. In this thesis, we see merit in combining depth correction/extension with the traditional problem of RGB-D SLAM.

CHAPTER IV

3D AWARE CORRECTION AND COMPLETION OF DEPTH MAPS IN PEICEWISE PLANAR SCENES

This section describes the depth enhancement technique needed for the proposed RGB-D SLAM method. Fig. 20 shows an overview of the overall system.

In this work, two images consecutively generated by an RGB-D sensor (MS Kinect) are taken as input, where both RGB and depth sensors are assumed to be calibrated, so their intrinsic and extrinsic parameters are estimated beforehand. Denote the RGB and depth images as $\mathbf{I}_c \in \mathbb{R}^{M\times N}$ and $\mathbf{I}_d \in \mathbb{R}^{M\times N}$ respectively. In the rest of the thesis, we denote $\mathbf{I}_d$ as the *raw* depth image, since it contains the unprocessed depth values that are directly measured by the sensor. Note that we do not limit the Kinect input to the reliable range (4 or 5 meters), but use all data provided by the Kinect sensor as raw input.

Refer to Fig. 21 for an example of a sample depth image taken in an indoor office scene. Clearly, not all the pixels in $\mathbf{I}_d$ have known depth values. These pixels are shown in black. They arise due to several reasons, including limitations in the depth sensor (*e.g.*, IR signal decay with square distance for the Kinect or inability to measure near objects at depths less than 80cm), IR absorbing or reflective surfaces, etc. In fact, many methods that process RGB-D image pairs from the Kinect either record images where the scene is at a maximum reliable distance d_{rel} (usually taken to be 3-4.5 meters) from the sensor or discard all pixels whose raw depth values exceed d_{rel}. These pixels are deemed to have *unreliable* depth values. Discarding unreliable pixels limits the range and impedes the generality of these methods. This issue is more significant when general indoor scenes are considered, e.g. in open areas, office spaces, reasonably sized rooms,

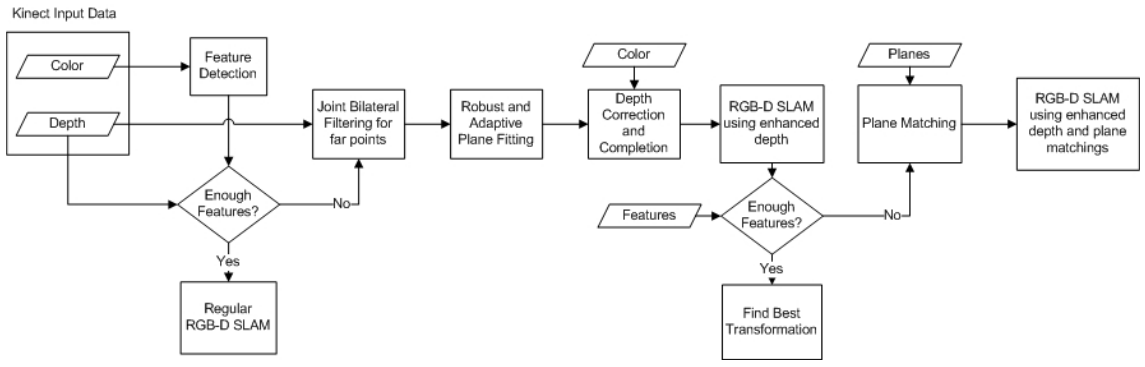
Kinect Input Data
Color
Depth
Feature Detection
Enough Features?
Yes
No
Regular RGB-D SLAM
Joint Bilateral Filtering for far points
Robust and Adaptive Plane Fitting
Color
Depth Correction and Completion
RGB-D SLAM using enhanced depth
Features
Enough Features?
Yes
No
Find Best Transformation
Planes
Plane Matching
RGB-D SLAM using enhanced depth and plane matchings

Figure 20: Overview of the proposed method. The input includes two Kinect frames with color and depth information. We perform feature matching and check if the number of features is sufficient. In cases where we have a low number of image features, we correct and complete the depth data, then perform RGB-D SLAM using the enhanced data. More features can be added by performing plane matching, which uses the first motion estimation.

museums, etc. In these cases, much of the scene is at a distance larger than the maximum reliable depth value d_{rel}. We show empirical evidence of this in Fig. 22, where we plot the average percentage of pixels discarded because they were deemed unreliable ($d_{rel} = 4.5$ meters) for a typical recording of a walk-through inside an office/lab environment. Discarding this large number of pixels in each depth image limits any subsequent processing or learning modules, including RGB-D semantic labeling and scene understanding methods. Interestingly, many of these unreliable pixels are projections of 3D scene points belonging to *objects* (*e.g.* floor, ceiling, walls, cabinets, etc.) that have extensions within the reliable range of depth pixels. Conceivably, it is of significant benefit to study how reliable depth values can be used to judge and even *enhance* unreliable depth values, when a particular prior model is assumed on the 3D scene. In fact, this could also be used to *complete* (or extend) the depth values beyond the raw measurements of the sensor (*i.e.* infer depth at pixels with unknown raw depth values). This thesis studies this problem for 3D planar scenes and proposes a novel framework that makes use of appearance and 3D cues from both $\mathbf{I}_c$ and $\mathbf{I}_d$ images to enhance and complete the raw depth. Planar scenes are valid descriptions of many indoor scenes containing man-made objects, *e.g.* building interiors, offices, homes, museums, etc. This scene prior has been used in the literature in the context of stereo vision [20], semantic labelling [28], and scene understanding [26].

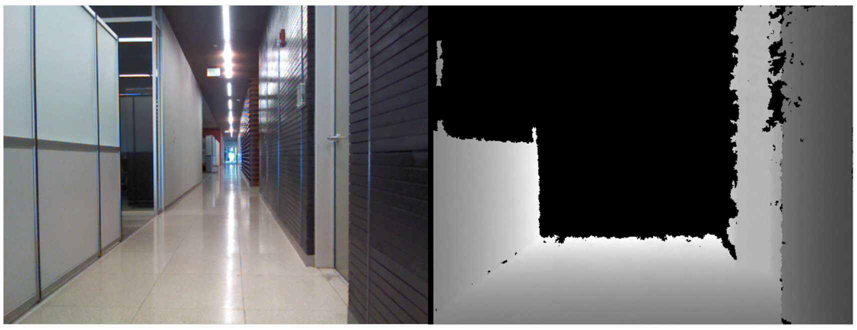
Figure 21: (left) Kinect RGB image. (right) Depth image. Black pixels do not have depth data.

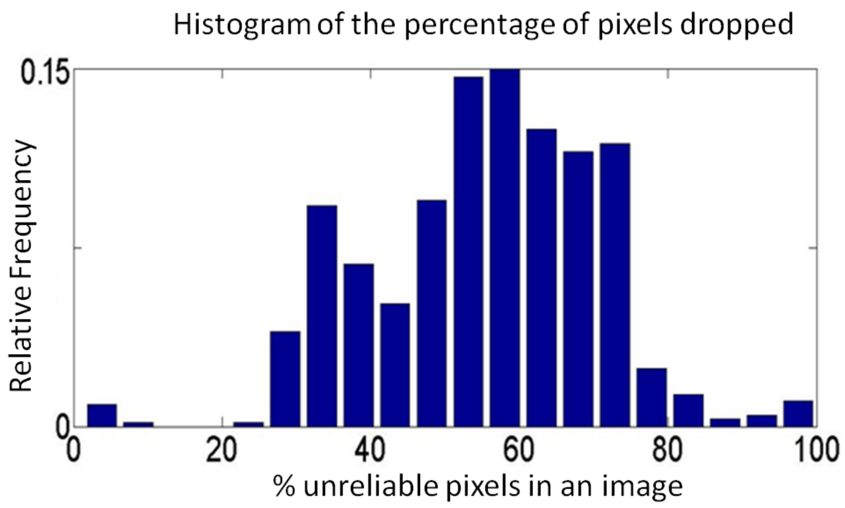

Figure 22: Histogram of the percentage of pixels discarded in a typical office/lab environment

A. Depth Measurement Noise:

The noise in Kinect depth measurements increases as the distance from the sensor to the object increases. For this reason, previous studies have omitted depth data at distances larger than 4 or 5 meters due to their unreliability. In this work, we try to maximize the number of features for which the depth data is available, so we try to make use of most of the data available from the Kinect. The accuracy of the depth data at all distances should be studied to assess its usage in any application. We provide empirical evidence for noise in depth measurements made by a typical Kinect sensor (refer to Fig.

23). Here, we plot the variance of the measured noise as a function of distance. This is done by fixing the Kinect as it images a single inclined plane (the floor). Multiple measurements are then taken for the same scene. Depth values at the same pixel in the depth image are then compared. Note that the noise is in the axial direction and noise at boundaries is not taken into account. A second degree ploynomial is fitted through the data, and the resulting axial noise can be modeled as

$$\sigma_d(z) = 2.481e^{-6}z^2 - 0.0025z + 1.228$$

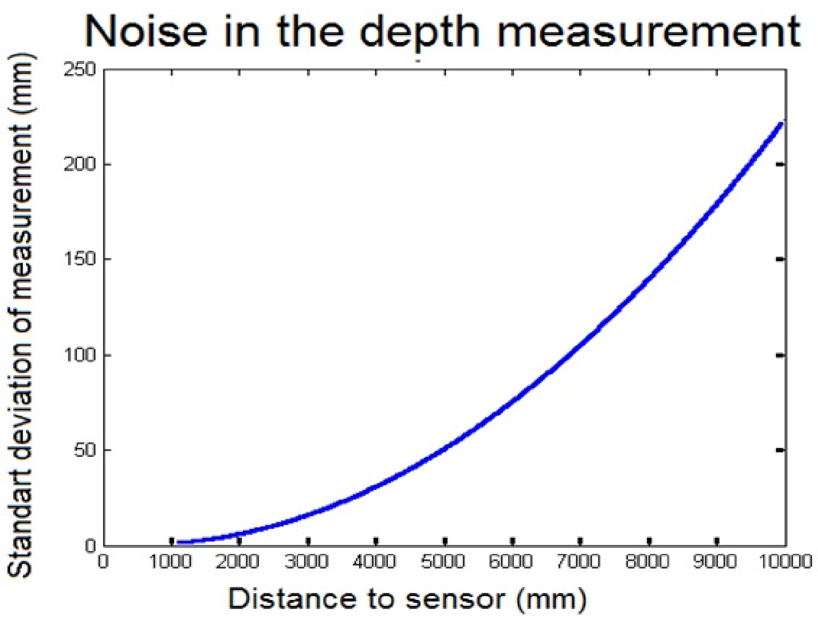

Figure 23: Plot of the standard deviation of the depth measurement as a function of distance to the sensor.

If each raw depth value at pixel **p** is considered a random variable, then this plot describes and quantifies its randomness as the distance to the sensor varies. In this work, we use a simple Gaussian error model to describe a depth value at **p**: $d(\mathbf{p}) \sim \mathcal{N}(\mathbf{I}_d(\mathbf{p}), \sigma_d(\mathbf{I}_d(\mathbf{p})))$. Equivalently, we model the depth random variable with a Gaussian distribution centered at the sensor's raw measurement and with a standard deviation that varies quadratically with depth. While Fig. 23 provides the value of $\sigma_d(\mathbf{I}_d(\mathbf{p})))$ for discrete values of depth, interpolation can be used to determine the error variance at arbitrary depths. Clearly, this error model oversimplifies the underlying imaging process, which is significantly affected by sensor limitations (*e.g.* the degradation of the IR signal with distance) and the scene itself (*e.g.* the orientation of a scene with respect to the image plane). However, we will see in the next section that this model lends itself useful in adaptively and robustly pre-processing the raw depth values.

B. Pre-Processing

Given a calibrated RGB-D sensor (Kinect), we aim to analyze its output data by looking at the whole range of depth values available. Applications that use devices like the Kinect disregard data outside a given range, due to unreliable nature of the sensor at these depths. To better understand the nature of these errors, we imaged a plane at 16 different depths using a Kinect, with 20 images taking at every location, totaling 320 frames. The distance of the plane to the sensor was varied from 1.2 meters to 10.2 meters in steps of 0.6 meters. Since the sensor is calibrated, we can estimate the 3D points of every depth image; we use these points to fit a plane and compute residual errors defined as the euclidean distance from each point to the best fit plane obtained. We want to look at the variation in measurement of the sensor as a function of depth, to do so, we compute a standard deviation of mean error at every plane location, and observe how this value changes as depth increases. Fig. 24 plots the change of standard deviation versus depth, it also shows a plot of a quadratic fit that best represents these values as a function of depth.

To better understand the effect of depth variations native to the sensor, Fig. 25 shows a depth image along with its corresponding 3D point cloud reconstruction. It is important to notice that, although it is not quite visible in the depth frame, the 3D point cloud representation of the scene is significantly distorted, particularly as depth increases. For example, note how the wall and ceiling points loose their natural shape. This is a direct consequence of the unreliable measurements made by the sensor at far

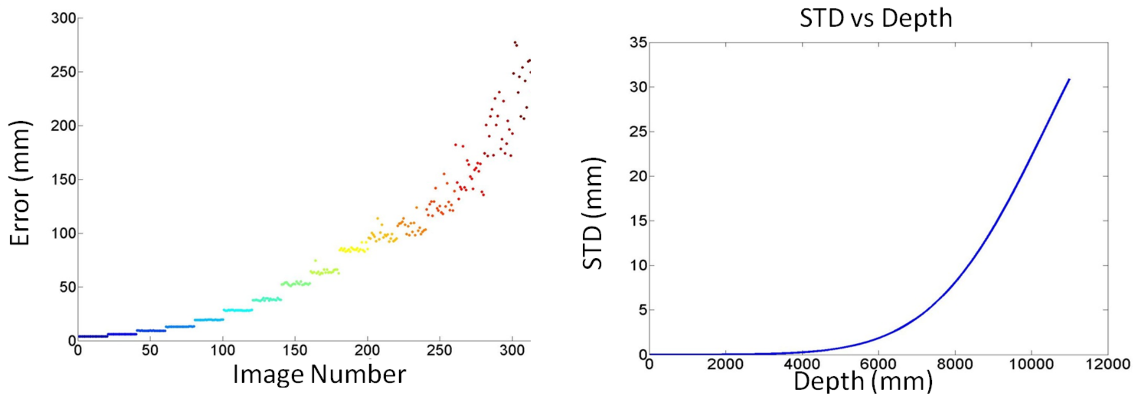

Figure 24: (left) Fitting error from depth data at different distances from sensor. Points of equal color are mean errors from different frames at same sensor distance; we notice the large variation or error as depth increases. (right) Standard deviation of error as a function of depth.

points.

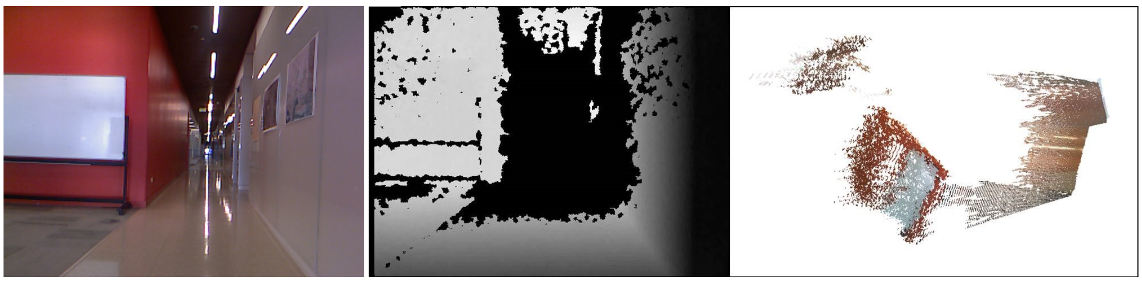

Figure 25: (left-to-right) RGB image, depth image, and the corresponding 3D view. Notice the high level of noise as shown in the 3D image.

1. *Adaptive Depth Smoothing:*

To reduce the effect of these 3D distortions, the projected 3D points are smoothed using the joint bilateral filter below. This filter makes use of both the depth image $\mathbf{I}_d$ and RGB image $\mathbf{I}_c$, as in [13]. We do not use this filtering method (that is unaware of the underlying 3D scene) to fill in unknown depth values in $\mathbf{I}_d$.

$$\hat{\mathbf{I}}_d(\mathbf{p}) = \frac{1}{k} \sum_{\mathbf{q} \in \Omega} \mathbf{I}_d(\mathbf{q}) F(\mathbf{p},\mathbf{q}) G(\mathbf{I}_d(\mathbf{p}), \mathbf{I}_d(\mathbf{q})) H(\mathbf{I}_c(\mathbf{p}), \mathbf{I}_c(\mathbf{q}))$$

In this filter, the smoothed depth at pixel $\mathbf{p}$ is a positive weighted sum of the raw depth values in the neighborhood Ω around $\mathbf{p}$. Each weight is a product of three similarity measures between $\mathbf{p}$ and its neighbor $\mathbf{q}$: within-image spatial closeness (defined by F), similarity in raw depth (defined by G), and similarity in appearance (defined by H). Similar to other methods, we take these three functions to be Gaussian. Using the previous sensor noise model, we model $G(\mathbf{I}_d(\mathbf{p}), \mathbf{I}_d(\mathbf{q}))$ as a Gaussian function $\mathcal{N}(\mathbf{I}_d(\mathbf{p}) - \mathbf{I}_d(\mathbf{q}), \sigma_d(\mathbf{I}_d(\mathbf{p})))$, where the standard deviation is depth dependent and obtained from the plot in Fig. 24. The advantage of such an approach is to make the bilateral filter adaptive to varying depths, which specifically allows for more suitable smoothing at large depths. Fig. 26 compares results of applying a joint-bilateral filter with and without depth-adaptive sigma values. In the latter case, we use the same Gaussian parameters for all pixels at all known depths. Note how the smoothing is less effective at large depths when the adaptive option is disabled.

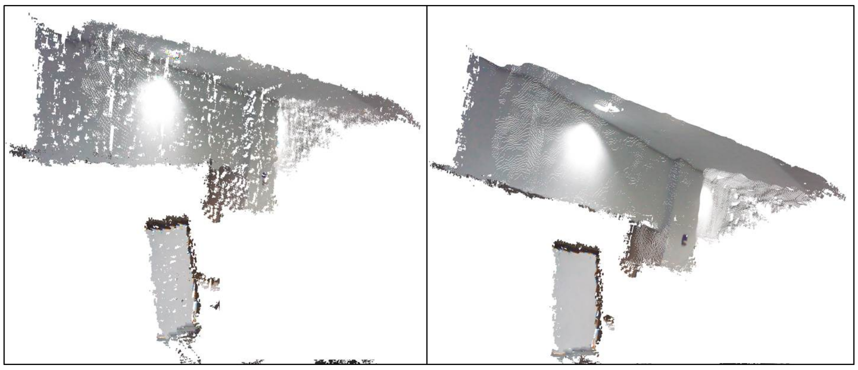

Figure 26: The result of the adaptive depth smoothing. The initial frame is shown to the left and the smoothed one to the right.

2. Adaptive and Robust Plane Fitting:

After depth-adaptive smoothing, we aim to detect and fit 3D planes through the 3D projections of pixels in $\hat{\mathbf{I}}_d$ with known depth. Similar to other methods, we use RANSAC for robust plane fitting; however, the criterion for a pixel to be an inlier to a fitted plane model (*e.g.*, having the distance of its 3D projection to the plane be less than

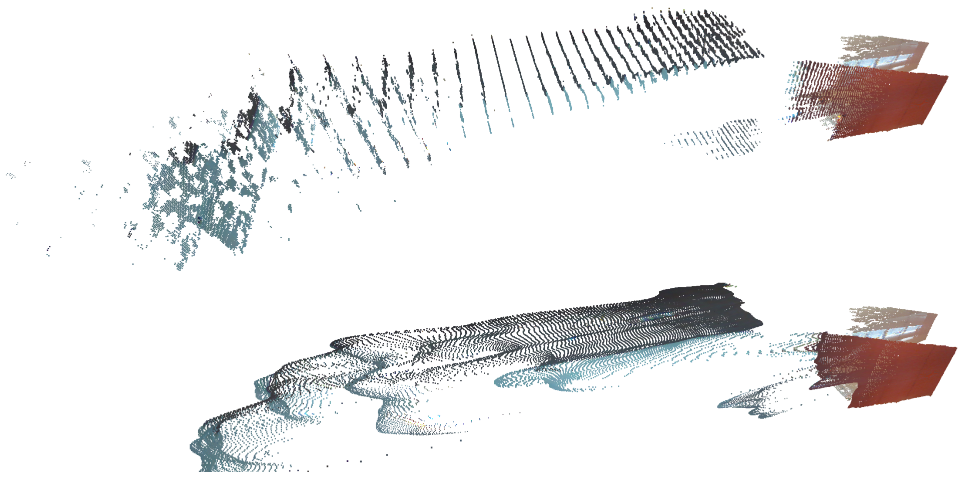

Figure 27: (Top) A 3D view of a frame before smoothing. Notice the discontinuities in depth, especially at large distances. (Bottom) The smoothed depth data. As can be seen, the discontinuities were removed but the final result is not the correct representation of the 3D world.

a threshold) should incorporate the noise in the sensor's depth measurements. Here, we note that we start the RANSAC process at pixels with smoothed depth less than d_{rel} before involving the farther pixels. We model the actual depth $d(\mathbf{p})$ at pixel $\mathbf{p}$ probabilistically as a Gaussian centered around the depth value $\hat{\mathbf{I}}_d(\mathbf{p})$ with a depth-varying standard deviation $\sigma_d(\hat{\mathbf{I}}_d(\mathbf{p}))$. Since the depth camera is calibrated, the 3D point corresponding to $\mathbf{p}$ is computed as

$$\mathbf{x}(\mathbf{p}) = d(\mathbf{p})\mathbf{K}^{-1}\tilde{\mathbf{p}} = d(\mathbf{p})\mathbf{t}(\mathbf{p})$$

where $\tilde{\mathbf{p}}$ is $\mathbf{p}$ in homogenous coordinates and $\mathbf{K}$ is the camera matrix. It is easily shown that $\mathbf{x}(\mathbf{p})$ is a Gaussian random variable (centered around the observed 3D projection $\hat{\mathbf{I}}_d(\mathbf{p})\mathbf{t}(\mathbf{p})$). Similarly, the distance $D(\mathbf{x}(\mathbf{p})|(\mathbf{n}, n_0))$ between $\mathbf{x}(\mathbf{p})$ and a 3D plane parameterized by a unit normal $\mathbf{n}$ (pointing towards $\mathbf{x}(\mathbf{p})$) and offset n_0 also has a Gaussian distribution. In particular, we have

$$D(\mathbf{x}(\mathbf{p})|(\mathbf{n}, n_0)) \sim \mathcal{N}(\mu_D, \sigma_D^2)$$

where

$$\mu_D = \hat{\mathbf{I}}_d(\mathbf{p})\mathbf{n}^T\mathbf{t}(\mathbf{p}) + n_0$$

and

$$\sigma_D^2 = \sigma_d^2(\hat{\mathbf{I}}_d(\mathbf{p}))\|\mathbf{n} \circ \mathbf{t}(\mathbf{p})\|_2^2$$

and $\circ$ is the Hadamard product. As expected, the variance of $D(\mathbf{x}(\mathbf{p})|(\mathbf{n},n_0))$ increases with depth and varies with the relative orientation of the plane with the camera plane. By representing this distance probabilistically, we replace the usual RANSAC inlier condition $D(\mathbf{x}(\mathbf{p})|(\mathbf{n},n_0)) \leq a$ with $p(D(\mathbf{x}(\mathbf{p})|(\mathbf{n},n_0)) \leq a) \geq \theta$. The latter probability can be computed straightforwardly using the cdf of a Gaussian. In this work, we take $a = 2$cm and $\theta = 0.8$. Also, normals of the observed 3D projections $\hat{\mathbf{I}}_d(\mathbf{p})\mathbf{t}(\mathbf{p})$ can be used to refine the inliers. As a result, using our probabilistic rule allows a plane model $(\mathbf{n},n_0)$ that is fit with 3D projections from reliable pixels (*i.e.*, their depth values are less than d_{rel}) to extend into farther pixels with less reliable depth values. This extension would not be possible and plane fragmentation would ensue, if the usual RANSAC inlier condition is used instead. Once the proposed RANSAC fitting method converges to a set of 3D plane equations and corresponding pixel inliers, we project the 3D projections of the inliers unto their respective planes and update $\hat{\mathbf{I}}_d$ to reflect this projection. This point-to-plane projection changes the depth values acquired by the sensor and, in most cases, corrects their values when a piecewise planar scene is assumed. If a fitted plane exists such that the vast majority of projected points lie in the halfspace designated by its normal and the principal angle between its normal and the normal of the image plane is negative (clockwise), then this fitted plane is designated as the floor. A similar rule designates the ceiling plane if it exists.

C. Depth Completion as an MRF

As a result of pre-processing (adaptive smoothing and robust plane fitting), the raw depth image $\mathbf{I}_d$ is transformed into $\hat{\mathbf{I}}_d$ and each pixel in $\hat{\mathbf{I}}_d$ with a known depth value is given a label corresponding to the label of the fitted plane it belongs to. We denote the resulting label image as $\mathbf{L}_0 \in \{0, 1, \ldots, l\}^{M \times N}$, where the 0 label designates pixels of unknown depth, 1 the floor (if it exists), and 2 the ceiling (if it exists). In this section, we describe how the initial label image $\mathbf{L}_0$ is relabelled through an iterative process that makes use of appearance and 3D cues from $\mathbf{I}_c$ and $\hat{\mathbf{I}}_d$. We denote the label image at iteration k as $\mathbf{L}_k$. As we will see, a consequence of this process is the iterative extension/completion of each plane label and the constriction of the 0 label. In other words, pixels with unknown depth values (labelled 0) can be assigned a plane label, if deemed likely from an appearance and 3D reasoning point of view. We formulate the iterative relabeling process as an iterative *interactive graph cuts* problem, where regional (unary) and boundary (binary) terms are exploited to relabel all pixels in the image while enforcing the labels of pixels that already have plane labels.

1. Determining Background Pixels:

Clearly, not all pixels in $\hat{\mathbf{I}}_d$ are projections of 3D points that belong to the l fitted planes. Due to sensor limitations and scene structure, other planes might exist in the 3D scene but they are not imaged at all. To allow pixels not to belong to the l fitted planes, we construct a *background* label, denoted as $(l+1)$. Since no depth information exists for background pixels, we label them according to how their projection rays (3D rays connecting the pixels to the camera's optical center) relate to the fitted 3D planes, as follows. A pixel is given an $(l+1)$ label if **(i)** it cannot belong to any of the l fitted planes (*i.e.*, its projection ray does not intersect any of the planes) or if **(ii)** the intersection between its projection ray and each plane j is at least $d_{\max}$ far from the closest known 3D point of plane j. In our experiments, we take $d_{\max} = 1$ meter. Condition **(ii)** assumes that the farther away a point is from the observed points of all fitted planes, the more likely it belongs to the background label. Background pixels are labeled as $(l+1)$ and added to $\mathbf{L}_0$. Fig. 28 shows an example of pixels labelled as background in a sample depth frame after plane fitting.

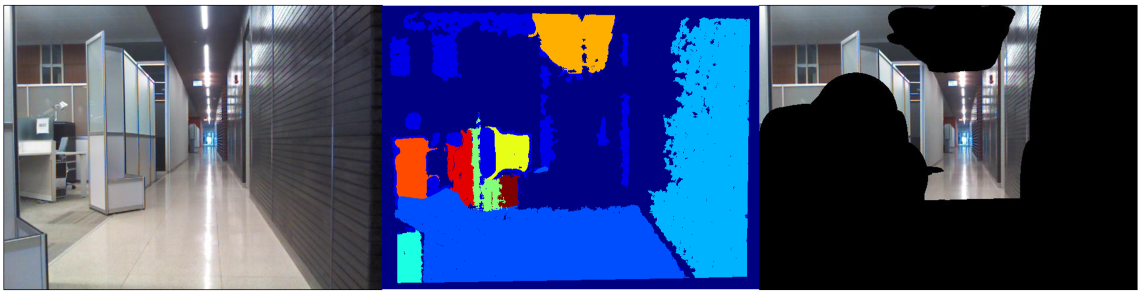

Figure 28: (Left-to-right) RGB image to be processed, initial labels, and ixels initially labeled as background.

2. Discriminative Appearance Model:

After determining a set of background pixels, we discriminatively model the appearance of each label (*i.e.*, pixels corresponding to 3D points belonging to the same fitted plane). All labelled pixels in the image are represented using a set of low-level image features that describe a pixel's color (local color histogram), neighborhood structure (HOG features), and texture information (LBP features). Using PCA, dimensionality reduction is performed to maintain 90% variance in the labelled data. A discriminative appearance model is formed by training a multi-class RBF SVM classifier on all labelled pixels. This model is a vector scoring function $h(\mathbf{z}) \in \mathbb{R}^{l+1}$, where $h_i(\mathbf{z})$ is the SVM score of labelling feature vector $\mathbf{z}$ as i, for all $i \in \mathscr{L}$ such that $\mathscr{L} = \{1, \ldots, l+1\}$.

3. Vanishing Lines:

Apart from appearance, other cues exist that shed light on the 3D structure of the scene. A widely used cue is the existence of vanishing line segments. This cue has been extensively used in scene recognition and understanding from single RGB images, especially in indoor piecewise planar environments [23]. In an image, vanishing points are usually extracted through a process of clustering line segments that vanish to the same point. However, in our case, we can explicitly compute certain vanishing points without any need for clustering or line detection. In planar scenes, plane segments tend to be perpendicular to each other, thus, many parallel 3D lines in the scene (belonging to the same or different planes) tend to be perpendicular to another plane in the scene. Therefore, we can easily estimate the vanishing point of 3D parallel lines that are perpendicular to fitted plane i by simply constructing at least two such lines (parallel to the normal of plane i) and projecting them unto the RGB image $\mathbf{I}_c$. We denote these vanishing points as $\mathscr{V}_1 = \{\mathbf{v}_i\}_{i=1}^{l}$, where $\mathbf{v}_i$ is the vanishing point of parallel 3D lines that are perpendicular to plane i. To generate other possible vanishing points, we use a method similar to [5] to obtain another set of vanishing points $\mathscr{V}_2$ that is disjoint from $\mathscr{V}_1$. We maintain a record of the clustered line segments that vanish to points in $\mathscr{V}_1$ and $\mathscr{V}_2$. Obviously, the process of extracting line segments and clustering vanishing points in $\mathscr{V}_2$ is prone to error, but it provides an additional 3D cue that can be harnessed in the relabeling process.

4. MRF Formulation

Given the label image $\mathbf{L}_0$, we now aim to label all pixels in $\hat{\mathbf{I}}_d$, especially those with unknown depth values. We model this labelling problem with a discrete Markov Random Field (MRF), where $\mathscr{L} = \{1, \ldots, l+1\}$ is the set of discrete labels, $\mathscr{P}$ is the set of all pixels, and $\mathscr{E}$ is the set of all connections defining local neighborhoods around

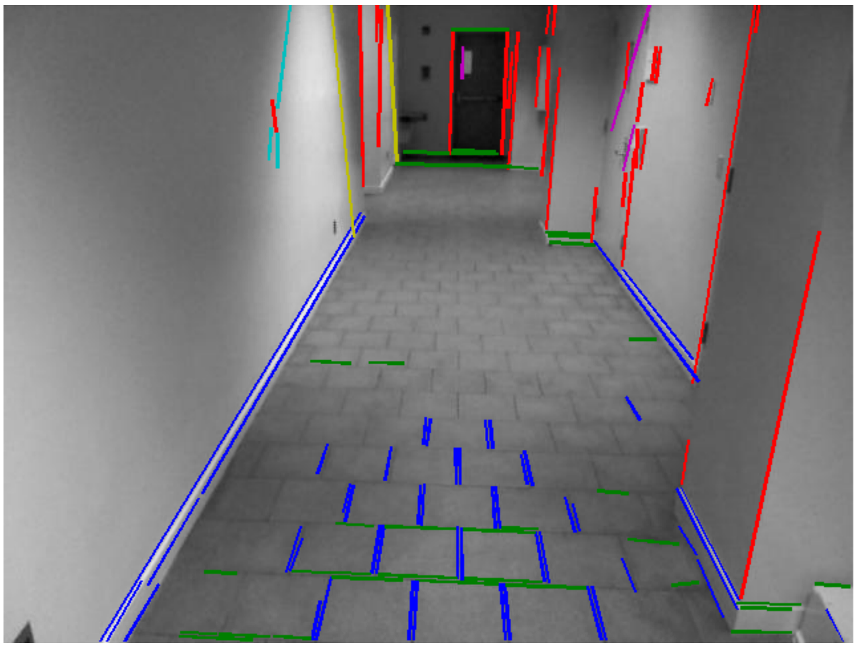

Figure 29: Vanishing lines in a corridor. Segments of the same color vanish at the same point

each pixel. In our experiments, we consider an 8-connected neighborhood. We seek a labeling $\mathbf{f}^*$ that minimizes the energy in Eq (1).

$$E(\mathbf{f}) = \sum_{\mathbf{p}\in\mathscr{P}} U(\mathbf{f}_p|\mathbf{p}) + \lambda \sum_{(\mathbf{p},\mathbf{q})\in\mathscr{E}} B(\mathbf{p},\mathbf{q}) \tag{1}$$

Here, $U(\mathbf{f}_p|\mathbf{p})$ defines the unary (or data) term, which quantifies the cost required to assign pixel $\mathbf{p}$ to label $\mathbf{f}_p \in \mathscr{L}$. Alone, this term treats pixels in $\hat{\mathbf{I}}_d$ independently, so a binary (or smoothness) term $B(\mathbf{p},\mathbf{q})$ is added for regularization, with tradeoff coefficient λ. This term quantifies the cost of assigning neighboring pixels $\mathbf{p}$ and $\mathbf{q}$ to different labels, *i.e.*, $\mathbf{f}_p \neq \mathbf{f}_q$. This energy can be minimized efficiently by $\alpha - \beta$ expansion iterations of graph cuts [8]. Since some pixels in $\hat{\mathbf{I}}_d$ are already assigned to fitted planes with non-background labels, we use a version of graph cuts (popularly known as interactive graph cuts) to guarantee that the labels of these pixels, after optimization, remain the same. This is sometimes referred to in the literature as enforcing hard constraints.

Unary (data) Term: Although the optimization technique (interactive graph cut) is well-known and has been used for various labelling problems in the past, the quality of the final labeling is mainly determined by how appropriate and informative the unary and binary terms are for labelling. In our case, the unary term is inversely proportional to the likelihood of a pixel belonging to a fitted plane. This term compares the appearance of a pixel to the discriminative appearance model of each plane and prevents label assignments that are incompatible in 3D under the piecewise planar assumption. In general, we set $U(i|\mathbf{p}) = -h_i(\mathbf{z}(\mathbf{p}))$ for each pixel $\mathbf{p} \in \mathscr{P}$ and each label $i \in \mathscr{L}$. This assumes that points belonging to the same plane in a planar scene look similar, which is usually a valid assumption. In what follows, we use 3D cues (from both $\hat{\mathbf{I}}_d$ and $\hat{\mathbf{I}}_c$) to regularize the labelling further.

To enforce the hard constraints, we follow a similar strategy as in [8], where $U(i|\mathbf{p}) = K \gg 0$ for each $i \in \mathscr{L} \backslash \{l+1\}$ and $\mathbf{p}$ such that $\mathbf{L}_0(\mathbf{p}) = j \neq i$. This large cost K prevents a pixel that is already labelled in $\mathbf{L}_0$ to switch labels. This enforcement is done for all labels except background because the manner in which we selected background pixels might have lead to incorrect labels. Moreover, we set $U(i|\mathbf{p}) = K$ for any pixel $\mathbf{p}$ whose projection ray does not intersect plane i. Making use of the piecewise planar assumption, we enforce that all intersections between projection rays and planes occur *above* the floor plane (if it exists) and *below* the ceiling plane (if it exists). This prohibits label assignments that lead to 3D points, which tunnel into the floor or pierce the ceiling. For these pixels, we set $U(1|\mathbf{p}) = K$ and/or $U(2|\mathbf{p}) = K$. Lastly, the unary term should not lead to label assignments that are contradictory in 3D. As such, we set $U(i|\mathbf{p}) = K$ for any pixel $\mathbf{p}$, which belongs to a line segment that vanishes to $\mathbf{v}_i \in \mathscr{V}_1$. This discourages $\mathbf{p}$ from belonging to a fitted plane, if it lies on a line perpendicular to that plane. In this case, the points belonging to the intersection of two perpendicular planes will be assigned to only one of the two planes and not both.

Binary (smoothness) Term: To allow for interactions between neighboring pixels in the labelling process, we define a binary term $B(\mathbf{p},\mathbf{q})$, which encourages label smoothness among pixel neighbors that have similar appearance and/or that are likely to belong to the same plane in 3D. In general, we set $B(\mathbf{p},\mathbf{q}) = \exp(-\Delta_c(\mathbf{p},\mathbf{q})\Sigma_c^{-1}\Delta_c(\mathbf{p},\mathbf{q}))$, where $\Delta_c(\mathbf{p},\mathbf{q}) = \mathbf{I}_c(\mathbf{p}) - \mathbf{I}_c(\mathbf{q})$. The covariance matrix Σ_c is estimated from $\mathbf{I}_c$. Moreover, we make use of vanishing points to discourage pixels lying on the same vanishing line segment to belong to different planes. So, we set $B(\mathbf{p},\mathbf{q}) = K$ when pixels $\mathbf{p}$ and $\mathbf{q}$ belong to the same line segment that vanishes to a point in $\mathscr{V}_1 \cup \mathscr{V}_2$. In our experiments and as suggested in [8], we set $K = 1 + \max_{\mathbf{p}\in\mathscr{P}} \sum_{\mathbf{q}:(\mathbf{p},\mathbf{q})\in\mathscr{E}} B(\mathbf{p},\mathbf{q})$.

5. Iterative Solution:

Once all unary and binary terms are computed for all pixels and labels, we solve the labeling problem using graph cuts [8]. Upon convergence, we use the final labeling $\mathbf{f}^*$ to determine the depth value of each pixel with label $\mathbf{f}_p^* \in \{1,\ldots,l\}$. This is done by intersecting the projection ray of $\mathbf{p}$ with fitted plane $\mathbf{f}_p^*$. The depth of a pixel labeled as background (*i.e.*, $\mathbf{f}_p^* = l+1$) remains unknown. Effectively, the original label image $\mathbf{L}_0$ has been relabelled to produce a modified version $\mathbf{L}_1$. Many pixels that had unknown depth values in $\mathbf{L}_0$ have been assigned depth values in $\mathbf{L}_1$, which allow for conformity to appearance models of existing planes and non-contradictory 3D layouts in piecewise planar scenes. In fact, the relabeling process can be rerun with $\mathbf{L}_1$ replacing $\mathbf{L}_0$. Obviously, the plane equations can be refined, the appearance model for each label needs to be retrained, and the unary and binary terms should reflect the changes in hard constraints. The resulting iterative relabeling process reaches convergence at iteration k when the change in label image $\delta(\mathbf{L}_k,\mathbf{L}_{k-1})$ is negligible. As such, our proposed approach corrects (through robust plane fitting and projection) and completes (through appearance and 3D aware relabeling) raw depth values recorded by an RGB-D sensor in a piecewise planar scene. In Fig. 30, we show how the end-to-end approach applies to a sample RGB-D pair of images.

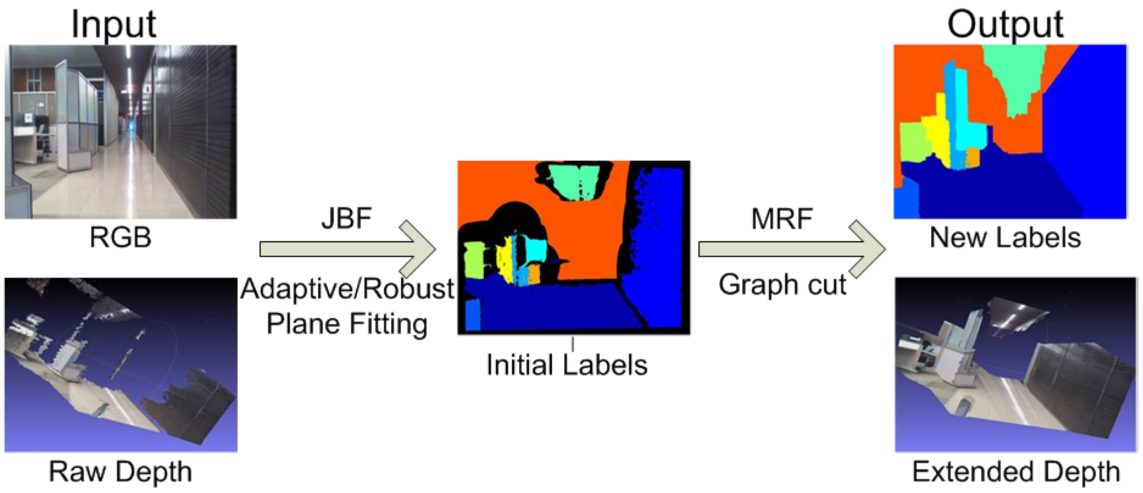

Figure 30: End-to-end process: We start with an RGB-D image pair. After segmenting planes, labels, and background label, we get a initial set of labels. The initial labels are fed to the Graph Cut solver. The final output is a complete set of labels, which are converted to new depth values. We can observe how the depth range is extended from the original 3D point cloud to the final result of our correction/completion process.

CHAPTER V

RGB-D SLAM IN FEATURE-POOR PLANAR ENVIRONMENTS

After enhancing the input depth data, we use it for SLAM. The proposed SLAM method comprises of two steps: an estimation step and a correction step. Note that the modification of the input is two fold. First, the input depth data for reliable and unreliable depth data is now corrected to improve the accuracy of the SLAM method. Second, 3D points previously discarded for not having depth data can now be utilized.

In the first step, we use a modified version of the RGB-D mapping of Henry [24] to find the optimized relative transformation that best aligns two consecutive frames. Since added data can be erroneous, outliers are removed using RANSAC while inlier contribution to the transformation relies on the score of the depth point added. The solution suggested by [24] is summarized in Eq (2).

$$\begin{aligned}\mathbf{T}^* =& \underset{\mathbf{T}}{\operatorname{argmin}}\left[\alpha\left(\frac{1}{|A_f|}\sum_{i\in A_f} w_i|\mathbf{T}(f_s^i)-f_t^i|^2\right)\right.\\ &\left.+(1-\alpha)\left(\frac{1}{|A_d|}\sum_{j\in A_d} w_j|(\mathbf{T}(p_s^j)-p_t^j).n_t^j|^2\right)\right]\end{aligned} \tag{2}$$

In the estimation step, we drop the second term which is regarded as the ICP term. In the ICP process, points that are closest to each other are matched and the registration minimizes the distances between them. When the frames needed to be registered are not the same (as in our case due to motion), the ICP method requires to reduce a certain percentage of matching points. Since the estimation of the percentage of points to be dropped needs motion approximation, we choose to drop the second term. Setting a constant percentage of points to drop is applicable in cases where the sensor motion does not change abruptly, but we aim at considering the more general case. Moreover, in areas such as corridors, minimal 3D variations exist, so the ICP solution has multiple local minima. Minimizing the first term is done via an adaptive RANSAC that incorporates sensor noise in selecting the inliers. RANSAC alignment first finds visual feature matches between two frames. Here, we use SIFT features but our proposed algorithm can be used with any other feature, especially since most feature detectors perform poorly in texture-less environments. RANSAC repeatedly samples three pairs of matches between two consecutive frames, finds the best transformation using Horn's method, and then evaluates the transformation. The evaluation is done by counting the number of inliers, i.e., the number of features correctly matched after the transformation. Inliers are chosen according to a threshold criterion that takes into account the sensor noise at all distances. After finding the transformation that yields the highest number of inliers, all inliers are used to find the optimal transformation. Not all inliers contribute equally to the final transformation.

$$w_g = \frac{MRFscore - minMRFscore}{maxMRFscore - minMRFscore}$$

MRF scores are used to determine the weights as shown above. MRF scores of added feature points are only considered. If the number of inliers is higher than a threshold, we skip the correction step, and the transformation obtained is the one used for motion estimation.

When the number of feature points is not high enough, we turn our attention to using the planes previously found. In order to find matching planes between two frames, planes found in the first frame are shifted using the estimated motion into the second frame. Similar planes are then detected and marked as matched planes. Matching points and planes are then used to find the optimal transformation as in [39].

CHAPTER VI

EXPERIMENTAL RESULTS

In this section, we will evaluate the effectiveness of our method in enhanceing/correcting and completing depth measurements obtained by a Kinect RGB-D sensor. For a quantitative comparison between the depth maps generated by our method and the raw depth maps obtained from the sensor, we generate a large-scale 3D point cloud of a typical indoor scene, which we will use to generate ground truth depth maps. Moreover, we show that our method can be useful in applications that make use of RGB-D data, including SLAM for self-tracking and depth upsampling from higher resolution RGB.

A. Dataset Compilation

To create the ground truth set, we scanned a large indoor area using 2 Kinect sensors mounted on a moving platform. The devices were pointed at different fields of view to avoid possible interference of their light patterns. The relative movement of the platform was constrained to a fixed translation along a predefined direction perpendicular to the floor normal, and a rotation of $\pm 90^\circ$ around the floor normal; such restrictions make the registration between frames trivial. A total of 700 frames were recorded, corresponding to 220 meters of stretch inside a typical office space.

Depth frames can be converted to 3D data using the calibration parameters of the sensors. In order to ensure data accuracy, we select the depth values on each frame that lie within the 800 mm to 4000 mm reliable range of the sensor. The final point cloud size was of around 63 million points. Fig. 31 shows 2 different views of the final reconstruction, with colored data embedded.

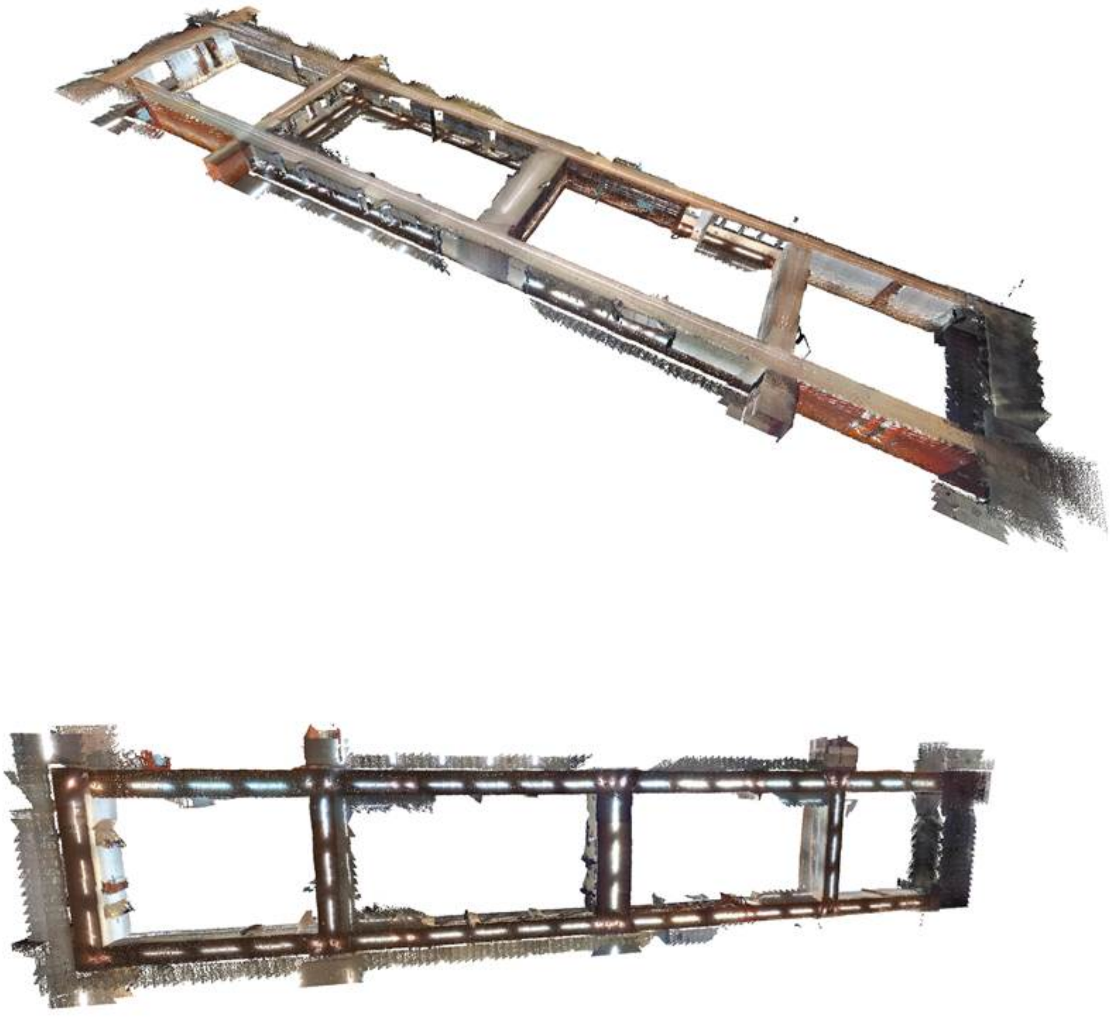

Figure 31: Different views of the large 3D reconstructed dataset. The total physical area covered is 220 meters, which yielded a point cloud of around 63 million points.

To complete the set, we need to create ground truth depth frames at each of the 700 locations, this is achieved by positioning the 3D points at the local frame coordinate system, and back-projecting them to the image frame using the camera intrinsic parameters; note that the back-projection process will create several 3D candidates for each pixel, but given the scene to camera visibility constraints, the ambiguity is solved by selecting the point closest to the image frame. The depth value at each pixel is the Z coordinate of the projected 3D point. Fig. 32 presents a triplet of color, raw depth, and ground truth depth corresponding frames.

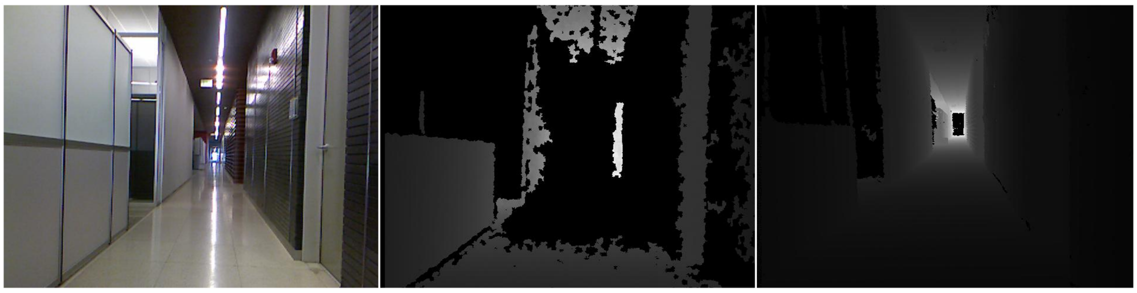
Figure 32: Left to right: RGB, raw depth, and ground truth depth. We notice the large amount of points missing in the raw depth image, due to the sensorâĂŹs inability to process large dark areas and reflective surfaces. The ground truth depth frame shows a complete version of the view by back-projecting the large 3D point cloud aligned to this frameâĂŹs view.

B. Quantitative Comparison

Using the ground truth set, we can test the accuracy of our approach and compare it to the raw data provided by the Kinect. We test our technique on 106 randomly chosen frames from the ground truth set. The result of our application is a new set of depth frames that improve the raw Kinect data in 2 ways, first by correcting depth values, particularly at large depths, and second by increasing the number of available depth pixels. In order to test for the first hypothesis, we compare the enhanced and raw depth with the ground truth 3D points. The comparison is done by converting the depth values of the enhanced and raw frame into 3D points, and calculating euclidean distances between closest points of these frames and the 3D point cloud of the scene. In order to do this computation in an efficient manner, we build a K-D tree for the ground truth point cloud once, and query the tree using the 3D points of each frame. Fig. 33 shows a plot of errors for both the raw depth data of the RGB-D sensor and our approach. Since the ground truth data was taken from raw depth values below 4 m, we expect the error in this range to be less in the raw depth frame of the sensor, but for depth value outside the reliable range of the Kinect (as pictured in Fig. 32), our approach improves the accuracy of the measured data by an average of 0.5 m.

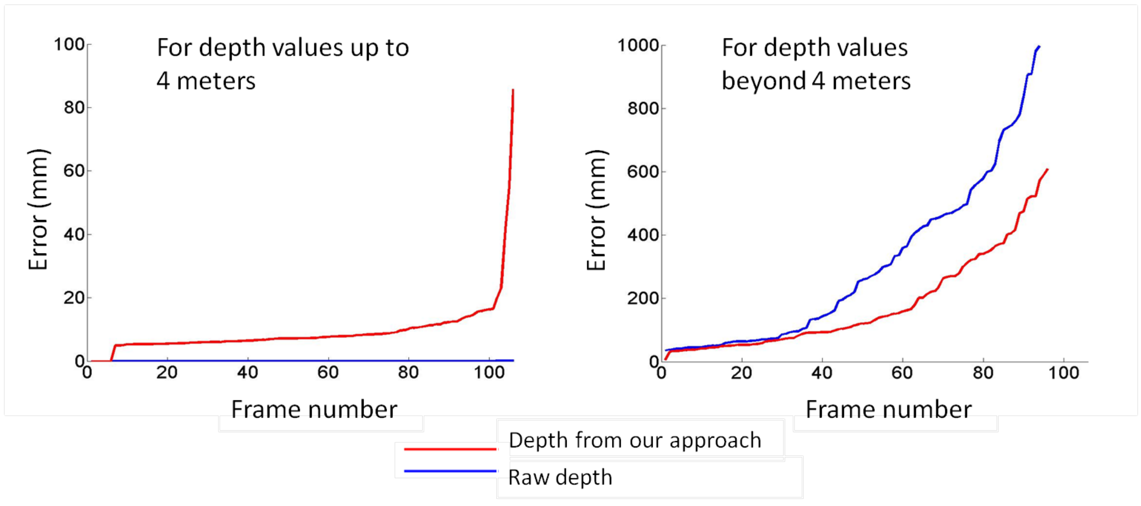

Figure 33: Comparison of raw depth values and depth values generated by our method respect to the ground truth for a large-scale piecewise planar scene. The left plot shows the errors at pixels of depth below 4 meters. Since the ground truth at depth lower than 4 meters was taken from the raw depth image, we expect the error to be zero in the blue curve. However, the power of our method is evident in the right plot where we show significantly lower error at depth values larger than 4 meters. This improvement comes from a combination of correction and completion of the raw depth.

Since our approach not only corrects but also extends the depth range of the sensor data; and thus increasing the number of depth pixels available, it is important to look at the average pixel increase as an additional measure of performance. Fig. 34 shows a plot of percentage pixel increase per frame, calculated as the percentage of pixels added by the our correction/completion with respect to the original raw data available. The mean increase by applying our correction/completion method is around 40000 pixels, with standard deviation of around 20000 pixels. It should be noted at this point that although an increase is present in more than 80% of the cases, the amount of increase always depends on the scene being processed, thus resulting in high standard deviation values.

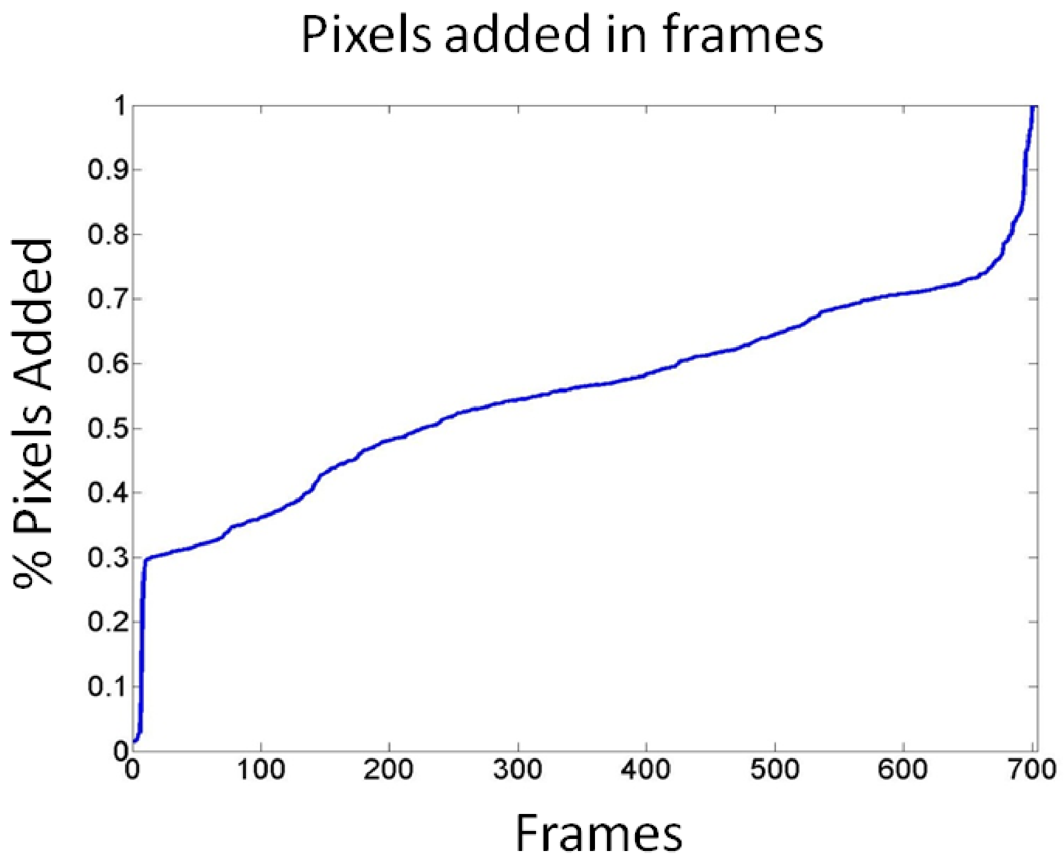

Figure 34: Percentage of depth values added to a single image. The proposed method is adding an average of 50% of depth pixels, and sometimes as much as 80%.

1. Result on NYU Dataset [37]

In addition to testing our proposed method on the set we compiled, we also make use of images from the popular NYUv2 dataset [37]. We choose images of piecewise planar environments (such as corridors or hallways) and apply our method on the raw depth images. We compare our corrected and completed depth map to that generated by the hole-filling (colorization) method used in [37]. We show two examples in Fig. 35, where we render 3D views of the raw depth data and the enhanced depth obtained by both techniques. Since the colorization technique is unaware of the 3D structure of the scene, depth points that were added by this method were not constrained to belong to any plane, leading to significant errors.

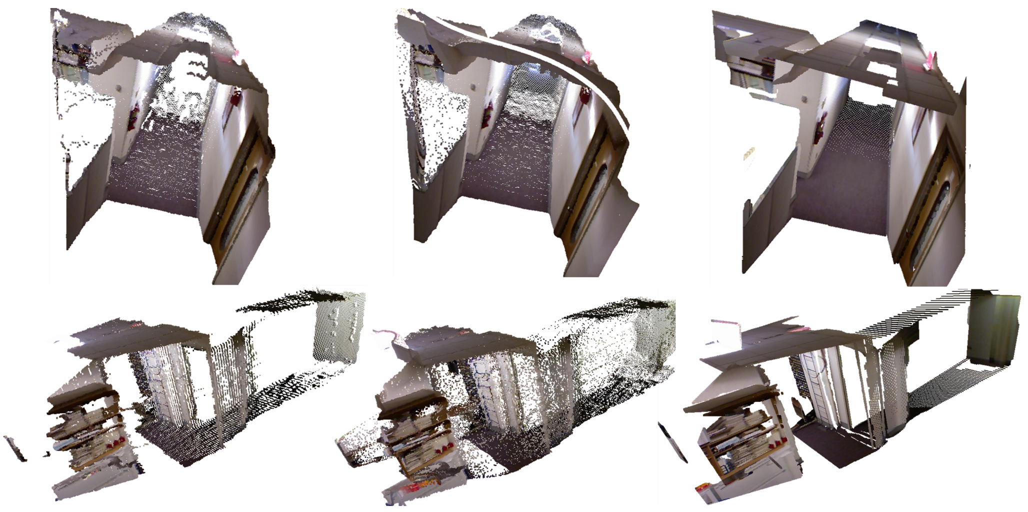

Figure 35: Left to right: 3D renderings of the raw depth, depth after applying the colorization method in [30], depth from the proposed method. Notice the noise in the depth data added by the colorization method. On the other hand, our method provides 3D aware depth correction and completion and a more realistic rendering.

C. RGB-D SLAM

In order to evaluate the effectiveness of our method, we make use of two corridor datasets. The first dataset is constituted of a 25 meters corridor with around 7 million depth points (increased to 10 million upon completion). For the first dataset, we show a qualitative result and compare it to using well-known RGB-D SLAM methods. The second dataset includes ground truth, so quantitative analysis is also performed. The second dataset has around 10 million points increased to about 15 million upon completion.

We first perform RANSAC alignment on the original depth frames and also on the corrected/extended depth produced by the method in Section C.. We compare the number of inlier points found in each case, as shown in Table 1. Clearly, our proposed method succeeded in increasing the number of inlier features significantly. This is due to the high number of depth points that were added but were previously discarded due to range limitation, reflection, or steep inclination. Moreover, adding the plane matchings to the pool of feature points that need to be registered serve as an addition of features. The average number of matching planes in our experiments was 3 (floor and side walls).

Alone, these planes were not enough for a proper registration as in [39] as they belong to the degeneracy case (Normals of these planes do not span the whole 3D world).

Table 1: Average number of inlier feature matches

	before enhancement	after enhancement
Dataset 1	3.47	41.36
Dataset 2	8.62	19.53

In Fig.36 and Fig. 38, we consider the first dataset. We show results of applying regular RANSAC based RGB-D SLAM method on this scene, as compared to applying our own SLAM approach using depth correction and extension. One can easily spot the drift when not enough feature points are detected. Moreover, the correction and extension provided a more complete 3D reconstruction by adding more 3D points while still maintaining the structure of the scene.

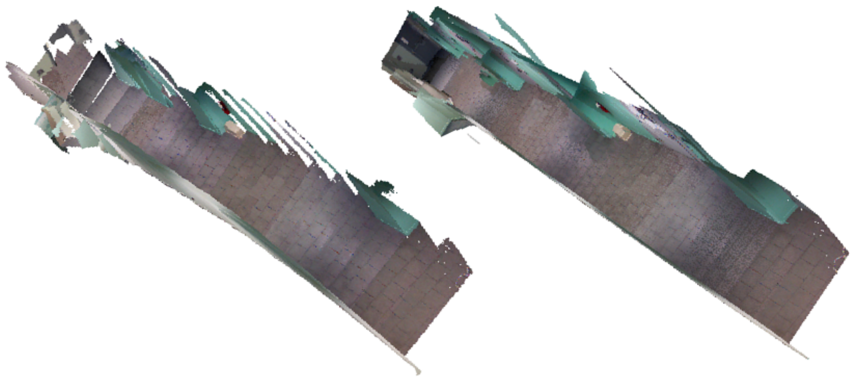
Figure 36: (left) SLAM results using the original depth information provided by the Kinect, (right) SLAM results using our proposed method

Fig. 37 shows a 3D view of the reconstruction of the whole loop of a corridor upon applying the proposed RGB-D SLAM method.

Fig. 39 shows the SLAM results for the second dataset. Not only did the proposed technique decrease the drift, but it also added a significant number of 3D points. The ceiling did not appear in the original reconstruction as it was more than 4 meters away. Since our method uses all Kinect raw measurements, we managed to fit a plane through the filtered data and reconstruct part of the ceiling too.

Figure 37: Dataset 2 whole loop results

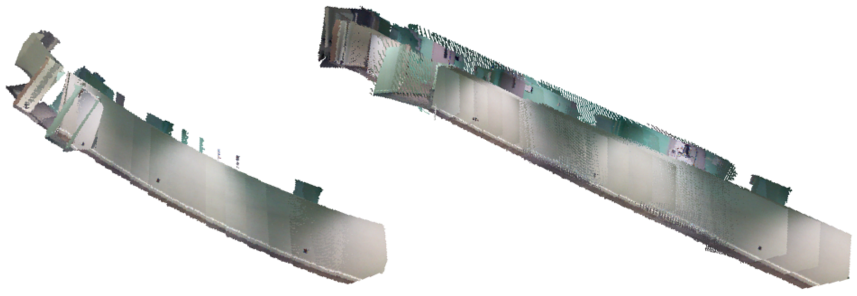

Figure 38: Another view of Fig. 36

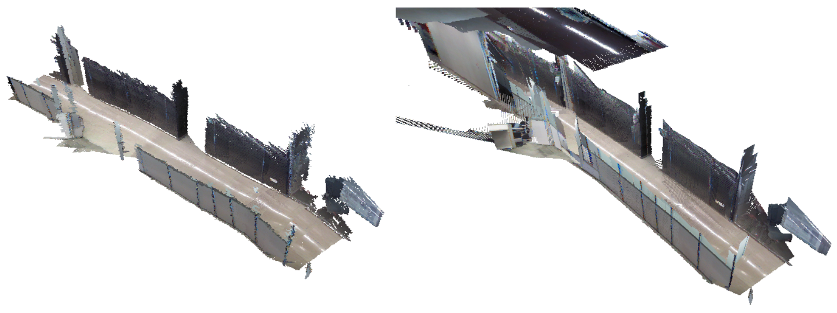

Figure 39: Dataset 2 results: (left) SLAM results using the original depth information, (right) SLAM results using our proposed method

Quantitative Comparison In order to perform a quantitative comparison, we apply our method on a dataset whose ground truth is known. The dataset was compiled by fixing the Kinect on a mobile platform which was moved throughout the hallway while maintaining the same motion. We compare our results to the ground truth data by measuring two errors: the drift in the sensor motion and that of the 3D point locations. The error in 3D point locations is computed as the distance between the predicted 3D point location and the true location. The predicted point location is calculated after movements of 2 meters throughout the hallway. Results are shown in Fig. 40. The average error in point locations is 440 mm when using the raw depth and 183 when using the corrected depth data. For the translation, the average drift is 53 mm for using raw data and 17.4 mm when using the corrected ones. Fig. 40 shows the difference in trajectory between the two methods. These results clearly validate the effectiveness of using the corrected/extended depth data generated by our approach.

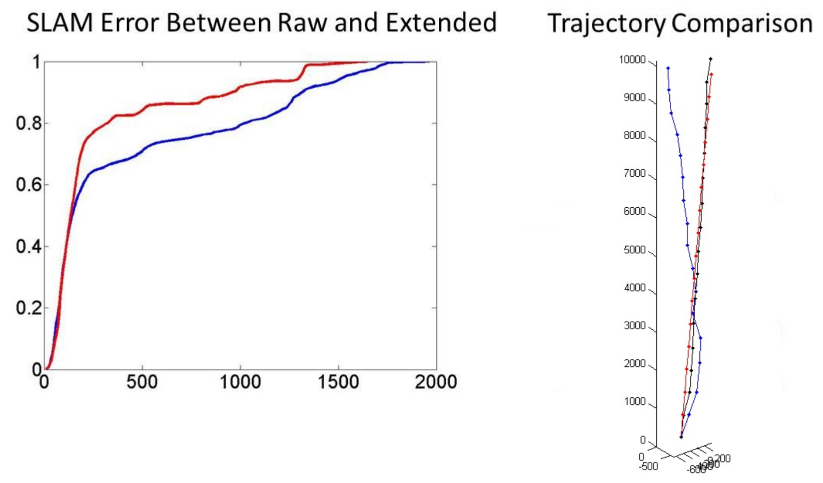

Figure 40: (left) Cumulative SLAM error when using raw (in red) and corrected/extended (in blue) depth data. Errors of our method average less than the errors incurred by using raw depth. (right) The ground truth trajectory (red), raw-depth SLAM trajectory (blue), our trajectory (black).

D. Depth Upsampling

Given that our 3D data is now represented as a combination of planar structures, it is easy to upsample the depth frame by looking at a high-resolution RGB image calibrated with the RGB-D sensor. Using the camera parameters of the high-resolution device, we can shoot rays that go from the center of projection, and through the image pixels. These rays may intersect one, many, or none of the planes available in our 3D set. We look at the set of intersections of the current ray and select the one with the smallest distance to the corresponding plane points, given that this distance is smaller than a threshold T = 10 mm. To test this application, we calibrated added a webcam that provides images at 1024x1280 resolution to the RGB-D system. The results is a higher resolution depth image that represents the same depth data as the corrected/extended frame. Fig. 41 shows the corresponding 3D reconstructions of the 2 frames. The upsampled point cloud provides similar scene information with a larger point set.

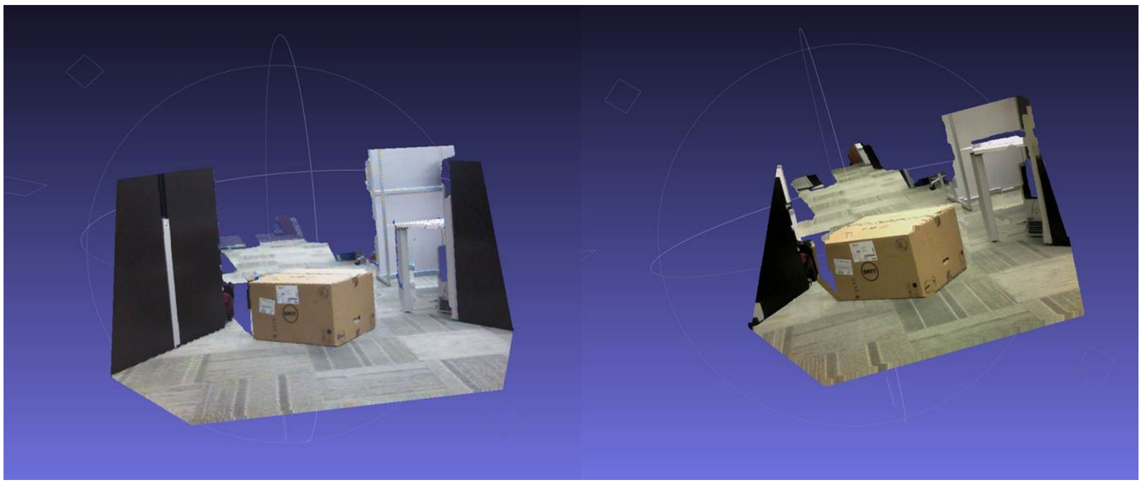

Figure 41: (Left) 3D points from completed depth with low resolution RGB image. (Right) 3D points from completed depth with high resolution RGB image

CHAPTER VII

CONCLUSION AND FUTURE WORK

In this thesis, a novel method was presented to use the complete set of depth values provided by an RGB-D sensor to better represent indoor piecewise planar environments and to improve the performance of RGB-D SLAM. By properly analyzing the sensor error at large depth, the given depth data was corrected, and planar labels were segmented from the scene. By applying the proposed method on a new large-scale ground truth data set, the new framework provides more accurate depth maps, having a larger number of pixels than those recorded by the RGB-D sensor. A new RGB-D SLAM method which makes use of the enhanced depth maps and plane features is also proposed. SLAM results on two datasets show a significant improvement to previous methods.

Future work include the addition of depth data for objects repeating in the scene based on their appearance. Moreover, detecting planar objects using edge detection techniques, which can make use of the high sensor noise at edges, can also increase the number of depth points in the scene.

BIBLIOGRAPHY

[1] Yoel Arieli, Barak Freedman, Meir Machline, and Alexander Shpunt. Depth mapping using projected patterns, April 3 2012. US Patent 8,150,142.

[2] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (5):698–700, 1987.

[3] Esra Ataer-Cansizoglu, Yuichi Taguchi, Srikumar Ramalingam, and Tyler Garaas. Tracking an rgb-d camera using points and planes. *ICCV*, 2013.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[5] JC Bazin and Yongduek Seo. Globally optimal line clustering and vanishing point estimation in manhattan world. In *CVPR*, pages 638–645, 2012.

[6] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[7] Jean-Yves Bouguet. Camera calibration toolbox for matlab. 2004.

[8] Yuri Boykov and Gareth Funka-Lea. Graph Cuts and Efficient N-D Image Segmentation. *IJCV*, 70(2):109–131, November 2006.

[9] Michael Calonder, Vincent Lepetit, and Pascal Fua. Keypoint signatures for fast learning and recognition. In *ECCV*, pages 58–71. Springer, 2008.

[10] Massimo Camplani and Luis Salgado. Efficient spatio-temporal hole filling strategy for kinect depth maps. *SPIE*, 2012.

[11] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[12] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[13] Sen-ching S Cheung. Layer Depth Denoising and Completion for Structured-Light RGB-D Cameras. *CVPR*, 2013.

[14] James Diebel and Sebastian Thrun. An application of markov random fields to range sensing. *NIPS*, 2005.

[15] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the rgb-d slam system. In *International Conference on Robotics and Automation*, pages 1691–1696. IEEE, 2012.

[16] Nikolas Engelhard and F Endres. Real-time 3D visual SLAM with a hand-held RGB-D camera. *.... of the RGB-D ...*, (c), 2011.

[17] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[18] David Fofi, Joaquim Salvi, and El Mustapha Mouaddib. Uncalibrated reconstruction: an adaptation to structured light vision. *Pattern Recognition*, 36(7):1631–1644, 2003.

[19] David A Forsyth and Jean Ponce. *Computer vision: a modern approach.* Prentice Hall Professional Technical Reference, 2002.

[20] Y. Furukawa, B. Curless, S.M. Seitz, and R. Szeliski. Manhattan-world stereo. In *CVPR*, pages 1422–1429, 2009.

[21] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[22] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[23] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. *CVPR*, pages 1849–1856, 2009.

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

[25] Gibson Hu, Shoudong Huang, and Liang Zhao. A robust RGB-D SLAM algorithm. *IROS*, pages 1714–1719, 2012.

[26] Zhaoyin Jia, Andrew Gallagher, Ashutosh Saxena, and Tsuhan Chen. 3D-Based Reasoning with Blocks, Support, and Stability. In *CVPR*, 2013.

[27] Johannes Kopf and MF Cohen. Joint bilateral upsampling. *SIGGRAPH*, 26, 2007.

[28] Hema Swetha Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Semantic Labeling of 3D Point Clouds for Indoor Scenes. In *NIPS*, pages 244–252, 2011.

[29] John J Leonard and Hugh F Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, 7(3):376–382, 1991.

[30] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH*, pages 689–694, 2004.

[31] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[32] Richard A Newcombe, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[33] Jaesik Park, Hyeongwoo Kim, Michael S. Brown, and Inso Kweon. High quality depth map upsampling for 3D-TOF cameras. *ICCV*, pages 1623–1630, 2011.

[34] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing icp variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.

[35] Joaquim Salvi, Jordi Pages, and Joan Batlle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827–849, 2004.

[36] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[37] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[38] Esdras soares de Medeiros Filho, Paulo Cezar Carvalho, and Luiz Velho. Coded structured light for 3d-photography: an overview.

[39] Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. Point-plane slam for hand-held 3d sensors. *International Conference on Robotics and Automation*, 2013.

[40] Roger Y Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *Robotics and Automation, IEEE Journal of*, 3(4):323–344, 1987.

[41] Liang Wang, Hailin Jin, Ruigang Yang, and Minglun Gong. Stereoscopic inpainting: Joint color and depth completion from stereo images. *CVPR*, pages 1–8, 2008.

[42] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994.

[43] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.