Mirror, Mirror on the Wall: Detecting Mirrors and Correcting Point Clouds

Sachit Mahajan
Carnegie Mellon University
sachitma@andrew.cmu.edu*

Abstract

Highly reflective surfaces found in common objects such as mirrors pose a major problem in reconstructing indoor scenes. Most of the current visual sensing methods cannot detect mirrors, which appear same as entrance-ways resulting in a severe performance degradation. This also causes significant errors in reconstruction and leads to incorrect mapping. This scenario becomes extremely problematic while robot exploration as it leads to the robots planning a path 'through' the mirror, causing a potential collision. We propose a method to detect such possible mirrors in the scene and then classify them as mirrors when seen at a better vantage point. This is done by attaching a fiducial marker to the sensing system and then using self-reflection to estimate the mirror plane. We also propose a novel coarse-to-fine region growing method to estimate the boundaries and extent of the mirror so that it can be tracked through the scene. Alternate methods are also explored and compared to the proposed approach. Additionally, we use this information to correct the point clouds by filling the mirror plane and re-projecting the points to correct position and gaining more information about the scene. We demonstrate this by reconstructing indoor scenes using a custom hardware setup where we fuse camera-Lidar data. The pipeline is designed to be generalizable such that it can be used with most visual and depth sensors. ***We detect the mirror, recover its position, orientation, and extent and correct the point clouds.***

1. Introduction

1.1. Motivation

Mirrors are common objects in indoor scenes, objects placed in front of plane mirrors have a corresponding image located behind the mirror. The distance from the image to the mirror is always identical to the distance from the object to the mirror. Current methods for 3D reconstruction cannot identify mirrors, since the beams are completely reflected, image formed behind the mirror is considered to be part of the scene and therefore mirrors appear the same as entrance-ways such as doors. Both active and passive scanning methods such as cameras, Lidars and depth sensors are fooled by these reflections. Existing indoor reconstruction methods do not acknowledge the problems caused by mirrors, rather either cover up mirrors while reconstruction or manually annotate and correct the point clouds later. There are also erroneous point-clouds in the Matterport-3D dataset [2] which have to manually selected and corrected by the users. While the 3D - reconstruction can still be fixed offline, robots performing SLAM and exploration are adversely affected by such mirrors. An example of incorrect pointclouds can be seen in Figure (1)

Figure 1. The image on the top is a test environment with two mirrors, the bottom images are a different views of the point clouds. It can be seen from the bottom middle point cloud that the points seen in the mirror are projected behind it making the reconstruction incorrect. This reconstruction had been done using Iphone 12 Pro (inbuilt Lidar) and 3D Scanner App. The image on bottom right shows the point cloud captured using a Velodyne VLP-16.

1.2. Proposed Approach

This project proposes a pipeline that can be used to automatically detect and identify mirrors in a scene by fusing visual and depth information. In visual data mirrors can

*The code will be open-source soon, some videos are available on the drive.

cause low-level color/texture discontinuities whereas jumps can be observed in depth. First, the 3D point cloud is converted into a 2D scan, similar to that given by a 2D Lidar and then we detect the discontinuities in depth. These discontinuities or jump-edges can be possible mirrors, a robot can use this data to plan a path to a better vantage point where we can classify the mirror. Once we are in front of the mirror we detect our own reflection. This is done by placing an inverted AprilTag on the scanning setup. These fiducial markers can be robustly identified and used to get a more geometric understanding of the scene. This allows us to robustly and accurately measure the mirror surface planar parameters and boundaries here onward referred to as mirror parameters (assuming mirror is planar or near planar), enabling us to track mirrors throughout the scene. Leveraging this information we can correct the reconstructed scene point clouds and gain more information about the scene. We see improvement in indoor exploration by robots as the biggest application of this project. We do not use any additional equipment than what is common for metric 3D reconstruction tasks, with the exception of the inexpensive fiducial marker. To our knowledge, this is the first complete pipeline that can be used by robots with generic scanning sensors to first locate possible mirrors and then use this knowledge to classify mirrors and correct the scene geometry.

To summarize the contributions of this work

- A method to detect jump edges in 3D point clouds which can correspond to possible mirrors in the scene
- A novel coarse-to-fine region growing method to estimate the mirror parameters- plane parameters and dimensions
- A method to use the mirror parameters to correct the 3D-reconstruction by adding the mirror as an obstacle and re-projecting the points seen behind the mirror to the correct position thus gaining more scene understanding
- A generalized pipeline where we combine the above mentioned steps and are also able to swap components so that the pipeline can work for different scanning setups
- A demonstration on real world data where we are able to detect mirrors and correct the point clouds.

2. Related Work

Given the extent of the issues caused by mirrors and reflective surfaces one would think that there would be multiple solutions to this problem. However, there has not been much research on this topic and to our knowledge no generalized solution exists yet.

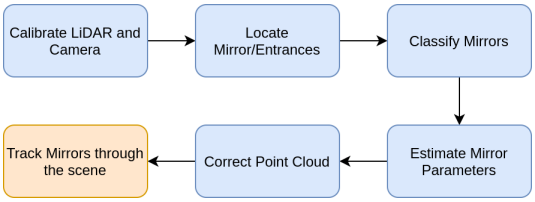

Figure 2. Our algorithm. The blue cells show what we implemented in the project, and the yellow cells show what we see as future work.

We were deeply inspired from the method presented in '**Reconstructing Scenes with Mirror and Glass Surfaces**' [13]. This work proposed a fully automatic pipeline that allows one to reconstruct the geometry and extent of planar glass and mirror surfaces while being able to distinguish between the two. The method robustly estimates the plane parameters of the mirrors and develops a multi-feature-based approach to detect boundaries of mirror surface. The authors have illustrated and visualized the multi-feature-based approach which gives a lot of intuition towards solving the problem and helped us define the heuristics used in Section 3.3. They also perform bundle-adjustment to improve their estimations. The authors also use a symmetric transform to correct the point clouds. The work uses a prototype 3D scanner, which implements a standard projector-camera system with an infrared pattern and truncated signed distance function (TSDF) based fusion. This is the closest paper to this project and we have used a lot of the findings from this work in formulation our pipeline.

One of the earlier solutions can be seen in the works [9] [14]. In [9] the authors incorporate mirror and glass detection in the localization, mapping and navigation frameworks. They fuse data from 2D laser scanner and from a different modality- an ultrasonic sensor to detect mirrors. Their approach is intended only for 2D scans to mark mirrors as obstacles in the occupancy map. The work is extended in [14], where they use a Gaussian model to predict the mirror after detecting discontinuities in the 2D Lidar scan. This is followed by an euclidean distance function to calculate the likelihood of the mirror for verification. They also incorporate and iterative closest point to match reflected points and find in the mirror. The last part of using ICP to find geometric symmetry is truly an innovative one and can be applied to our work as well. Since the authors were working only with a 2D scan the problem was easier. The objective of this work is to just to classify mirrors as obstacles in 2D scans for robot navigation. We want, however, not only to detect the fact that a mirror is present but also to recover its position, orientation, and extent and correct the point cloud in 3D scans which has further practical applications.

Other works exploit the multi-return properties of Lidars to detect glass and reflective surfaces. The authors in [3]

use a 2D multi-echo Lidar to get up to three echos with distances and intensities. They use the reflective intensity values to then filter out the mirrors. The very recent paper [16] uses a very similar setup to ours working with a Velodyne HDL-32E to detect mirror and glass surfaces using the dual return property. This work is also focused for robots performing indoor SLAM. Similar to our work they also re-project the points behind the mirror to the correct locations using householder transformations [6]. The end result of the work is same as ours to reject mirror points, correct them and treat mirrors as obstacles. They use the optical properties of different objects to build a reflection model about how laser light interacts with the glass and claim Lidar can estimate the distance of things behind the mirror with weakened intensity due to twice passing through the window. They perform an intensity peak analysis to detect the mirror plane. Since the 3D Lidar we used (VLP-16) has the same dual-return modes we tried to use some of the algorithms mentioned in the paper, we noticed that if the mirror was clean both the dual returns- last return and strongest return corresponded to objects behind the mirror. We did see a noticeable difference in intensity values which could be used to detect mirrors, however that would a significant amount of testing and tuning. This approach would also not be scalable and generalizable to other scanning methods including time of flight methods such as RGB-D cameras which are commonly used in indoor robots. One advantage this method had over ours was that the robot did not need to be in-front of the mirror to classify it.

The work [4] creates a tool to identify mirrors in 3D laser scans and to correct the corresponding 3D point clouds for robot navigation. They tackle the specific problem of detecting rectangular mirrors of known dimensions. They detect possible mirrors by making a 2D panoramic range image and detect jump edges and extract the 3D contours. They then apply a connected filter and use thresholds based on the prior dimension information to classify mirrors and then fit a plane using PCA. Contrary to this method we do not assume the mirror to be rectangular or of known dimensions. We do take some inspiration from the jump-edge detection, however we convert the 3D scan to a 2D scan not a range image to do so. This is discussed in detail in later sections.

Some other methods such as [15] and [5] use deep learning to segment mirrors and reflections. [5] focuses on removing reflections from scenes to prevent detection of false positives by exploring in detail how state of the art two-stage detectors suffer a loss of broader contextual features, and hence are unable to learn to ignore these reflection and then present an approach to fuse instance and semantic segmentation for this application, and subsequently show how they are unaffected by this loss of context, and are better able to learn to distinguish reflections from true positives. real world surveillance data with a large number of reflective surfaces. [15] presents MirrorNet a novel method to segment mirrors from an input image. They achieve this by constructing a large-scale labeled mirror dataset with a variety of daily life scenes. The authors then propose the novel network, MirrorNet, for mirror segmentation, by modeling both semantic and low-level color/texture discontinuities between the contents inside and outside of the mirrors. This method is then evaluated and shown to outperform the carefully chosen baselines from the state-of-the-art detection and segmentation methods. This network also fits perfectly in our pipeline, where we can detect mirror masks and use that to correct the point clouds. In our testing we found that MirrorNet works really well when there is one mirror in the scene but tends to fail when there are none or more than one mirrors in the scene. We still conduct some experiments incorporating MirrorNet with our pipeline which then allows us to classify mirrors and correct point clouds even when we are not exactly in front of the mirror.

There are prior solutions which detect mirrors using additional sensors or custom setup such as high speed time-of-flight sensors [11] or multi-signal time of flight sensors [3]. They rely on the fact that glass is a partially transparent and partially reflective surface provides valuable information since one can actually observe two distinct signals from the transmitted and reflected parts of the scene . We can also use structured light sensors and use the fact that mirror images are inverted. We initially tested a laser pattern emitter, however, due to imperfections in mirrors we received multiple patterns- the reflected one and the one mirror surface. This could be accounted for but then again in pursuit of a more generalized solution we did not further explore this option. The paper [1] employs a Projector-Camera system and deduces that the local frequencies in the captured pattern for mirror-like surfaces is different from the projected pattern. This property is then used to design a custom Projector-Camera system to segment mirror-like surfaces by analyzing local frequencies in the captured images.

There has been some research into reconstructing specular surfaces by using direct and inverse maps [8]. Such methods are more used to correctly reconstruct glossy, transparent or translucent objects.

After exploring all these works we gained inspiration and decided to design the algorithm introduced in the above section, and test the algorithm on a scene with mirrors and using a custom setup with a Logitech C920 and Velodyne VLP-16 3D-Lidar.

3. Methodology

In this section we will expand on the major steps that we implemented in our pipeline. A flowchart of the process we implemented is shown in Figure 2. Our hardware setup can

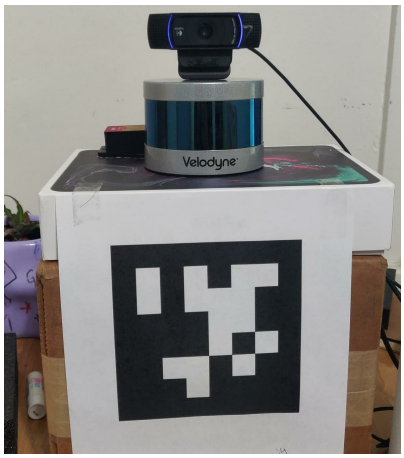

Figure 3. Our hardware setup with a Logitech C920 and Velodyne VLP-16 3D-Lidar.

be seen in 3.

3.1. Calibrate Lidar and Camera

We first calibrate the cameras using the camera_calibration package of ROS [10] and a checkerboard. Then we calculated the extrinsics of Lidar and Camera, this was done by using lidar_camera_calibration package to which we pass selected points in Camera Image and their correspondences in Lidar point cloud. The Lidar points are then projected in the Camera frame. We pre-process the point clouds by first converting the raw Velodyne data to an organized point cloud using the Point Cloud Library (PCL) [7]. We also extract the normals for point cloud by using clustering neighbouring points and fitting planes. We observed that the noise made planar surfaces have different normals, so we used edge-aware Bilateral filtering on the point cloud to smoothen the areas with smaller gradients while preserving the edges.

3.2. Locate possible mirrors

We have already established that mirrors and entranceways (eg. doors) appear the same, so we start with identifying these as possible mirrors. We do this by finding depth discontinuities we refer to as 'jump-edges'. Mirrors are usually placed on planar walls, in the Lidar scan this deviation in depth from planar surfaces can be found. We can use this to detect contours in 3D scans similar to [4]. We do this by compressing 3D scan to a 2D scan and detection deviations, by extracting the middle ring data from Velodyne scans. This can also be done by only using some rows if we are using a range image. Next, we start with two points and project them to wht world and derive the equation of the line. We then find the point-line distance of each subsequent point. If a point is close to the line we add that point and refine the line, this helps when the data is noisy. If a point is away from the line we consider it a possible edge start point and start the making a line using this point and the next point. We keep building a list of possible edge starts along with their line equations and the next time we detect a deviation in a point we find the distance from all the lines in the mentioned list. If the point is close to any of the lines in the list and the index are placed sufficient far away such that the previous deviation was not due to noise, but not far enough that the jump edge is too wide, we declare the possible jump points as a jump edge.

The significance of this step is that we can pass these possible mirror edges to the robots planner. This way the robot can plan a path such that we see the mirror is in front of the robot and we can use the steps explained later to determine the mirror parameters.

3.3. Classify mirrors and Estimate Mirror Parameters

Now that we know where the possible mirrors are we need to positively identify them. We use two methods to do so and discuss their pros and cons.

3.3.1 Method 1: Self Reflection and Region Growing

We have placed an inverted fiducial marker- AprilTag [12] on our setup. AprilTags can be robustly identified and accurately localized. Once the robot is in front of the mirror we can assume we will be seeing the AprilTag using which we can classify a mirror. We can refer to Figure (5) to visualize the scene. We know the point corresponding to AprilTags center P_{tag} and the use the apriltag package of ROS we get the point corresponding to center of AprilTag found behind the mirror plane $P_{tagimage}$. Using this information we can get a point on the mirror plane P_m which is the mid-point of the line joining real and virtual points P_{tag} and $P_{tagimage}$.

$$P_m = \frac{P_{tag} + P_{tagimage}}{2} \tag{1}$$

Now we need to estimate the normals of this plane. There are two ways we can do this

1. We know the dimension of the AprilTag seen (ω) and its center $P_{tagimage}$. The apriltag detection package estimates the 6dof pose of the tag, this includes the three rotation angles. We denote these angles as the Euler angles α, β, γ. Using is information we can now get 4 more points corresponding the 4 corner points of the AprilTag. We can now fit a plane to these five points and estimate the normals n.

2. Alternatively, we can define a normal of a plane antiparallel to us $n_p = (0, 0, -1)$ where we are in the coordinate frame of the camera with Z axis depicting the

depth and the negative to show that the plane is pointing towards the camera, and then use the rotation defined by Euler angles α, β, γ to rotate the normals.

$$n = R^T n_p \tag{2}$$

The next step is to estimate the extent of the mirror. We do this by a coarse-to-fine region growing method.
We first discretize the image into square grids. We now want to classify each grid as 'mirror' or 'not mirror'. We start by marking grids in which contain the AprilTag as 'mirror grids'. We add all neighbouring grids to a scoring queue which we incrementally process and repeat till adding their neighbours till the queue is empty. We score the grids based on the following heuristics

- **Self-Reflection**: If we can see any part of the AprilTag in grid it is marked as Mirror Grid
- **Depth greater than mirror plane and Depth Discontinuity**: As aforementioned, mirrors appear as depth discontinuities on an otherwise planar surface. If the grid has such discontinuities we assign it a higher score. If a grid contains points with depth greater than that of P_m we assign a higher score.
- **High Normal Variance**: We know the normals of the mirror plane, we can assume that mirror boundaries or the wall the mirror belongs to is parallel to it and has same normals. Two normal parallel to each other have

$$|n_{plane} \cdot n_{point}| = 1 \tag{3}$$

  we assign a score $w_{normal\_variance} * (1 - |n_{plane} \cdot n_{point}|)$ to all the points.
- **RGB Variance**: Mirrors can also be observed as discontinuity in textures and will have a higher variance in RGB values, this can however be counterproductive if the mirror-boundaries/walls are textured as they will have a high variance. Hence this parameter has a lower weight.

We now have a score

$$score = \Sigma w_{heuristic} * s_{heuristic} \tag{4}$$

and if the score is greater than aa set threshold we classify grid as a mirror.

All of the scores are normalized by the using the empirically observed maximum values in those categories. Now each time a grid gets classified as mirror its neighbouring grid is added to queue. When a grid is classified as not mirror it is marked as a boundary and it's parent grid is further divided in finer grids and all of them are reclassified. This enables a finer approximation of mirror boundaries.

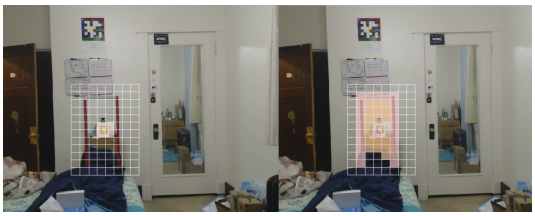

Figure 4. Left to Right shows an example visualization of region growing where we start with marking the grids containing AprilTag as mirror grids and then classifying neighbouring grids

For better understanding a visualization is given in Figure 4. We were not able to test this last coarse-to-fine step and is mentioned again in Future Work in Section 7.

We now have the mirror plane (M) parameterized as normals and a point on plane

$$M = [n, P_m] \tag{5}$$

Pros

- We can accurately identify the mirror and estimate it's parameters.
- Works with mirrors of any shape or size given that we can see our own reflection.

Cons

- We need to be in front of the mirror with the AprilTag in view.
- We can be limited by the number of Lidar points on the image, if scene is close some parts of image won't be covered and if the scene is far there will be too little points in each grid to decide.

3.3.2 Method 2: MirrorNet

We discussed in Section 2 about MirrorNet [15], which allows is to segment mirror using only visual data. The network is pretty fast and accurate when the scene has one mirror. We get the mask image and throw away all the points that lies inside this mask as mirror points.

Pros

- We can identify mirrors from all angles as long as they are seen in the image.

Cons

- Works well when there is only one mirror in the scene. When there are multiple mirrors only one is detected and when there are no mirrors it sometimes find mirrors where at wrong spots.

- We cannot estimate the mirror parameters using this technique as we don't have any information about mirror plane.
- Running the network is still slower and requires more resources

We can use a combination of both techniques where we use the first method to identify mirrors in front of us and use the second method when it is not. However, we cannot proceed to point cloud correction mention in Section 3.4 with the second method as do no have the mirror surface parameters.

3.4. Correct Point Cloud

By this step we have classified the mirror and have estimated the mirror parameters. The Figure (5) can be better used to explaining the convention in this section.

We have a point behind the mirror P_i, the mirror normals n and one Point on the mirror Plane P_m. We paramterize the plane as

$$ax + by + cz = d \tag{6}$$

where $n = [a, b, c]^T$ correspond to the normals and d can be found using the point P_m. The point P_m corresponds to the middle point of the line joining the AprilTag and the AprilTag image in the mirror. Now we can complete the plane equation by solving for d

$$d = -P_m \cdot n \tag{7}$$

We find out two point from P_i, the first being where it intersects with the mirror P_1 and the second P_2 being the correct position of the point P_i. We know that laser beam starts from origin P_o $(0, 0, 0)$ and ends at the point P_i (X_i, Y_i, Z_i). Using the equation of the plane we can figure out where line given by $P_o + \lambda P_i$ intersects with the plane $ax + by + cz = d$

$$\lambda(a * x_i + b * y_i + c * z_i) = d \tag{8}$$

$$\lambda = \frac{d}{a * x_i + b * y_i + c * z_i} \tag{9}$$

which leads us to

$$P_1 = \lambda P_i \tag{10}$$

We know P_i is the reflection of the actual point P_2 about the mirror plane, we can figure out P_2 by finding the reflection of the Point P_i about $ax+by+cz = d$. We do this by finding the point P_N which is the foot of the line perpendicular from P_i to the mirror plane, we can find P_N by

$$P_N = n * k + P_i \tag{11}$$

where

$$k = \frac{-n \cdot P_i - d}{|n^2|} \tag{12}$$

$$\implies k = \frac{-a * x_i - b * y_i - c * z_i - d}{a * a + b * b + c * c} \tag{13}$$

We can see that the Point P_N is the midpoint of reflected Pi and its corresponding real point P_2. Hence the coordinates of P_2 are

$$P_2 = 2 * P_N - P_i \tag{14}$$

We now have the a point P_1 which corresponds to a point on the mirror and we make it an obstacle, and P_2 which corresponds to the actual point that was reflected in the mirror hence giving us more information of the world.

Alternatively, we can also use subtract P_1 from P_i to shift the coordinate frame to a world where the mirror plane passes through origin and get the reflection using householder transformation

$$R(\hat{n}) = [I_{3\times3} - 2\hat{n}\hat{n}^T] \in \mathbf{R}^{3\times3} \tag{15}$$

this gives a symmetry transform that transforms a point into the reflected point as used in [13]

$$S(N) = \begin{bmatrix} R(\hat{n}) & 2d\hat{n} \\ 0^T & 1 \end{bmatrix} \tag{16}$$

4. Experiments

We use the setup seen in Figure (3). We test the our setup in a room with two different mirrors creating three test sequences, two with one of the two mirrors and one with both the mirrors. The test scene can be seen in Figure (1). We will perform experiments to test the steps in abovementioned sections.

4.1. Calibrate Lidar and Camera

Even after repeated attempts our calibration was not perfect, we have elaborated on this in Section 5. Figure (6) shows the Lidar points projected to camera image. The colors are just to represent the depth, blue to green to red shows increase in depth. The points in mirror are red signifying higher depth.

4.2. Locate possible mirrors

We use ranges from the middle ring of Lidar to simulate a 2D Lidar and detect jump edges. The possible mirror locations can be seen as red spheres in Figure (7).

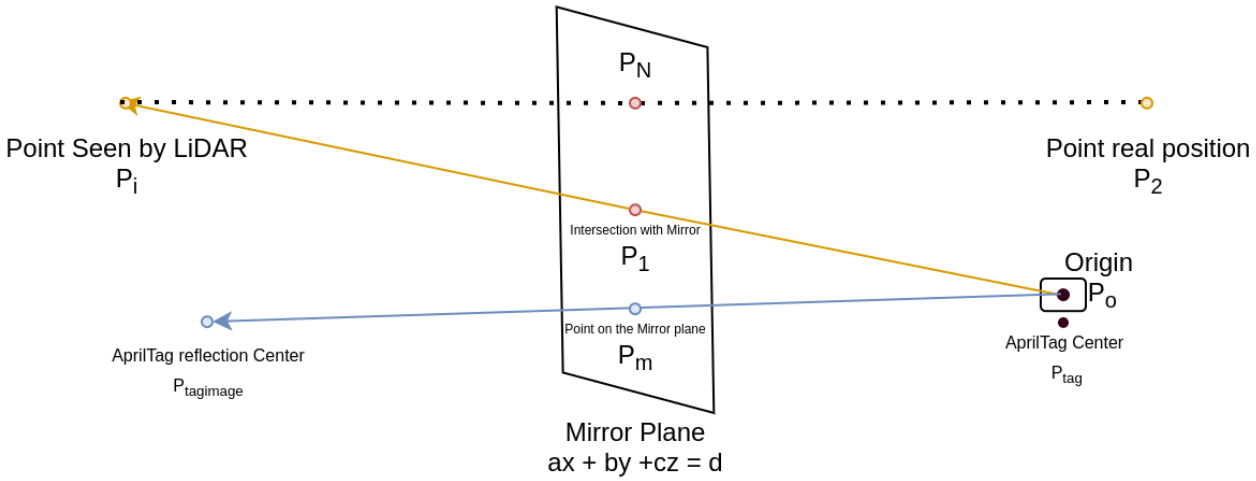

Figure 5. Reflection of point in a mirror

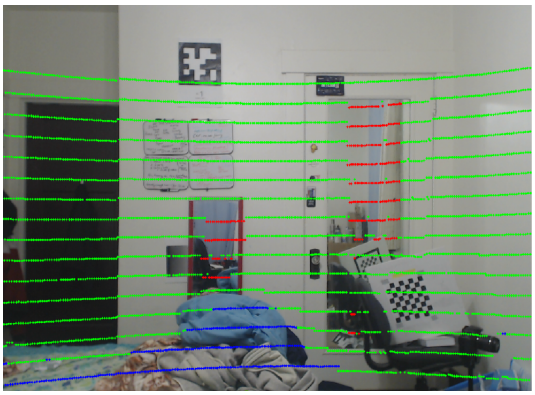

Figure 6. Lidar points projected to camera image. The colors represent the depth, blue to green to red shows increase in depth.

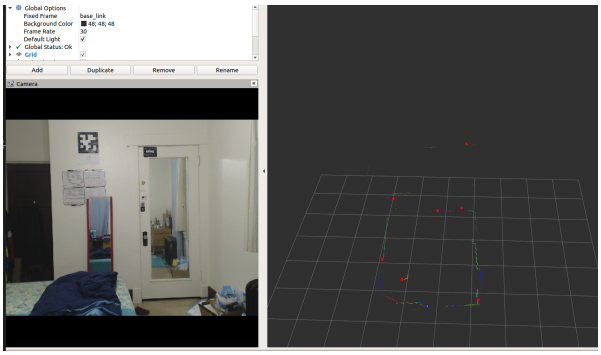

Figure 7. Jump edges in Lidar scan, the red spheres indicate presence of possible mirror

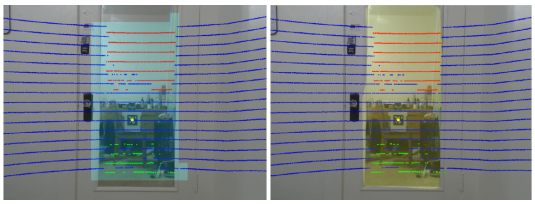

Figure 8. The image on the left has mask (blue) derived from region growing while image on the right has mask (yellow) obtained from MirrorNet

4.3. Classify mirrors and Estimate Mirror Parameters and Correct Point Clouds

Figure (8) shows masks obtained from both the methods. The region growing mask (left) accounts for error in calibration and considered additional grids but missing out on grids which had no Lidar points, while the MirrorNet method (right) covered the whole mirror.

Note: Due to error in extrinsics calibration and lack of time to tune heuristic parameters for the Region Growing Method (Section 3.3.1) we did not have good initial results, for the sake of project completion we add some bias to the grids which we knew already had the mirrors and added retro-reflective tape to mirror boundaries which helped reduce noisy measurements.

4.3.1 Method 1: Self Reflection and Region Growing

Figure (9) and (10) show the results on two different mirrors. We can see that not only is the mirror marked as obstacle, we also gained more points allowing us to know the scene better. A fixed square grid size of 32 pixels was use for region growing.

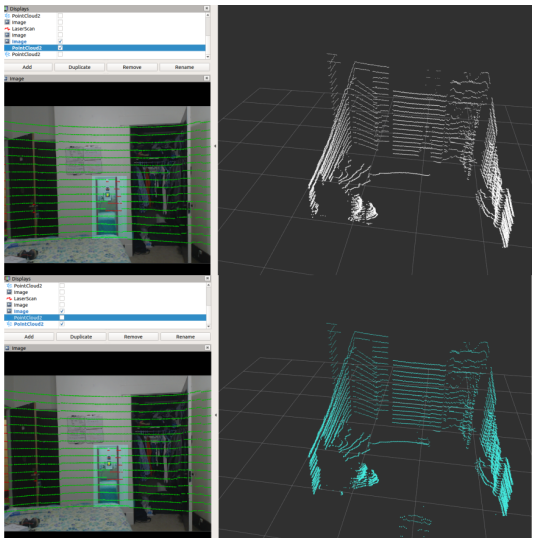

Figure 9. Top image shows input point cloud, and bottom one shows corrected points. The masks can be seen on images on left

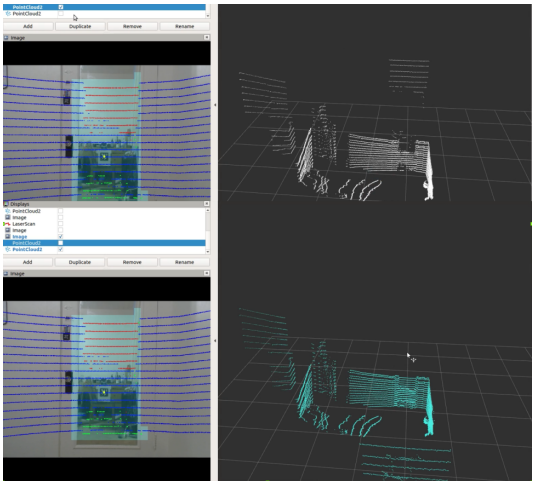

Figure 10. Top image shows input point cloud, and bottom one shows corrected points. The masks can be seen on images on left

4.3.2 Method 2: MirrorNet

Figure (11) shows the masks obtained from MirrorNet.

4.3.3 Combination of Both Methods

Figure (12) shows us the scene with both the mirrors. We can see that only the points corresponding to mirror in front of us have been corrected (Method 1), the points on the other method (Method 2) have been removed from the point cloud but not corrected as we have not estimated the mirror parameters.

Figure (13) shows that the points in-front of the mirror are not removed.

Figure 11. Masks obtained from MirrorNet

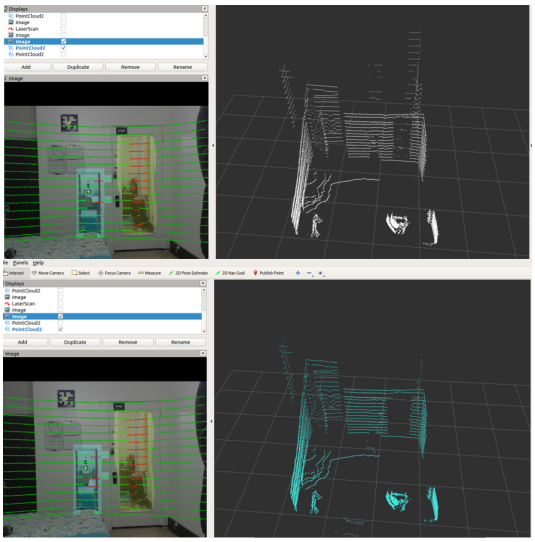

Figure 12. Top image shows input point cloud, and bottom one shows corrected points. The masks can be seen on images on left

5. Challenges Faced

1. The extrinsics calibration between camera and Lidar was not very accurate. We performed this calibration more than ten times however with unsatisfactory results. In retrospect, the cause of this error was probably the vibrations caused by motion of Lidar moving the camera. We had to hard-code some heuristics for mirror extent estimation to solve this problem.

2. Dust and imperfection of mirror caused incorrect estimates, sometimes it led to the reflection not being clearly seen or mirror not appearing as jump edge.

3. The lack of a localization system prevented us from moving our setup to improve our results and demonstrate some additional functionalities. We tried to place another AprilTag in the scene and attach an IMU to the setup, and used them to estimate our relative pose,

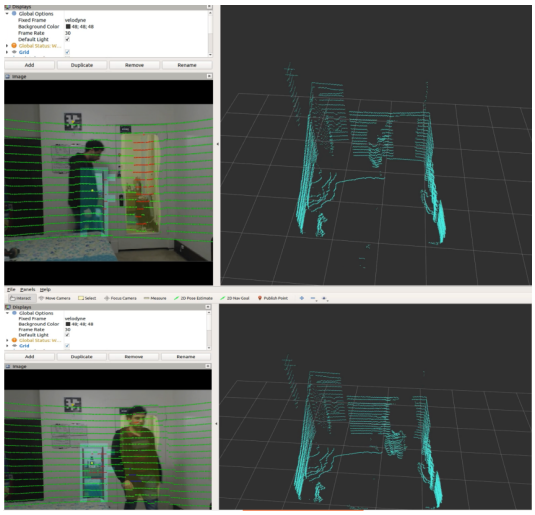

Figure 13. The points in front of the mirror are kept and not corrected/removed

however, this was very noisy and caused errors. We did not want to divert our attention to fixing these errors.

6. Conclusion

We propose a pipeline to overcome problems caused by mirrors in indoor reconstruction and robot navigation. Our method can be used to detect the mirror, recover its position, orientation, and extent and correct the point clouds. We also demonstrate this method on different indoor scenes with mirrors collected through our custom hardware setup. We also replace some steps of our pipeline with different methods and compare performance. This work acts a proof of concept that a novel method of detecting self-reflection and region growing can be used for mirror reflection detection and reconstruction correction in indoor scenes.

7. Future Work

We want to deploy this work on a robot where we can use the planning, localization and dense mapping capabilities. If we have a good SLAM system we can track mirrors through the scene and once mirror parameters are estimated we can correct point clouds even when we do not see the self-reflection. One way to do that is to keep warping the mirror mask we estimated based on our relative motion estimate. This way we only need to see the AprilTag in a single image and can then transfer the observation information to all other frames using the scene geometry and SLAM poses. Observing the mirror (and potentially the AprilTag) in additional frames may increase the accuracy of the estimated mirror plane and helps in accurately detecting the mirror boundary.

We explained the coarse-to-fine approach in Section 3.3.1, however, could only implement a coarse version. We can make further discretize the boundary grids into smaller grids and estimate better boundaries. This would enable better estimation while not significantly increasing computation complexity.

We have explained the multiple heuristics we used in Section 3.3.1. However, we did not get enough time to individually tune the weights and combine them together. We can visualize each of them individually so that we can understand them better and combine them. The paper [13] does a visualization of the multi-feature-approach to get more intuition.

We have the assumption that the mirror is planar or near planar, but we can use the points on the plane to fit a mirror surface even if it is planar. This way we can extend our algorithm to curved mirrors and add a curvature parameter.

One of our proposed method relies on seeing our own reflection at least once and needs an AprilTag on our setup. We can use our knowledge of scene geometry to classify reflection, for example if we have mapped the scene we can use point matching algorithms like ICP to find reflections by checking if same patterns appear twice and estimate the mirror parameters. This does require fairly accurate localization and mapping systems and a complex matching algorithms.

8. Acknowledgement

This work was carried out as the final project for 15-663 Computational Photography (Fall 2020) at Carnegie Mellon University. We thank Prof. Ioannis Gkioulekas for his advise and continuous guidance as well as generously providing the necessary equipment.

References

[1] Rajat Aggarwal and Anoop Namboodiri. Detection and segmentation of mirror-like surfaces using structured illumination. pages 1–8, 12 2016. 3

[2] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 1

[3] Rainer Koch, Stefan May, Patrick Murmann, and Andreas Nuchter. Identification of transparent and specular reflective material in laser scans to discriminate affected measurements for faultless robotic slam. *Robotics and Autonomous Systems*, 87, 10 2016. 2, 3

[4] P.-F Käshammer and Andreas Nuchter. Mirror identification and correction of 3d point clouds. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5/W4:109–114, 02 2015. 3, 4

[5] David Owen and Ping-Lin Chang. Detecting reflections by combining semantic and instance segmentation, 2019. 3

[6] Rui Rodrigues, Joao P Barreto, and Urbano Nunes. Camera pose estimation using images of planar mirror reflections. pages 382–395, 09 2010. 3

[7] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. 4

[8] Silvio Savarese and Pietro Perona. Local analysis for 3d reconstruction of specular surfaces — part ii. volume 2351, pages 759–774, 05 2002. 3

[9] Shao-Wen Yang and Chieh-Chih Wang. Dealing with laser scanner failure: Mirrors and windows. In *2008 IEEE International Conference on Robotics and Automation*, pages 3009–3015, 2008. 2

[10] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. 4

[11] Andreas Velten, Thomas Willwacher, Otkrist Gupta, Ashok Veeraraghavan, Moungi Bawendi, and Ramesh Raskar. Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging. *Nature communications*, 3:745, 03 2012. 3

[12] J. Wang and E. Olson. Apriltag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198, 2016. 4

[13] Thomas Whelan, Michael Goesele, Steven J. Lovegrove, Julian Straub, Simon Green, Richard Szeliski, Steven Butterfield, Shobhit Verma, and Richard Newcombe. Reconstructing scenes with mirror and glass surfaces. *ACM Trans. Graph.*, 37(4), July 2018. 2, 6, 9

[14] Shao-Wen Yang and Chieh-Chih Wang. On solving mirror reflection in lidar sensing. *Mechatronics, IEEE/ASME Transactions on*, 16:255 – 265, 05 2011. 2

[15] Xin Yang, Haiyang Mei, Ke Xu, Xiaopeng Wei, Baocai Yin, and Rynson W. H. Lau. Where is my mirror? *CoRR*, abs/1908.09101, 2019. 3, 5

[16] Xiting Zhao, Zhijie Yang, and Sören Schwertfeger. Mapping with reflection - detection and utilization of reflection in 3d lidar scans. *CoRR*, abs/1909.12483, 2019. 3