Image processing and computer vision

student number: 1924931

1. Introduction

In this experiment, I have completed all tasks about the 3D reconstruction of several spheres. Initially, I randomly generated six spheres in a 3D space and captured these spheres using two cameras positioned at different angles, projecting them onto the camera plane. Subsequently, I applied the Hough Circle Transform to both images to obtain the parameters of these circles. To accomplish 3D reconstruction, it was essential to determine the corresponding spheres' centers and radii. I identify corresponding centers in the two images by using epipolar constraints, and then reconstructed these centers in 3D space. For the radii, for each circle I selected a specific point on the circle and reconstructed it in 3D space using the same method used for the centers. By calculating the distance between the center and this point, I determined the radius and thereby completed the 3D model reconstruction. To verify accuracy, I projected this reconstructed model back onto the planes of the two cameras and compared it with the ground truth. Finally, I also use noisy relative pose between the cameras to 3D Reconstruction again and compare with the previous results.

2. Method

2.1 Hough Circle Detection

I utilized the cv2.HoughCircles function to perform the Hough Transform and detect circles in both view0 and view1. This function accepts multiple parameters, crucial among them are those that determine the accuracy of circle detection. The 'minDist' parameter represents the minimum distance between the centers of detected circles. I set this to “view0_blur. shape [0] / 10”, meaning the minimum distance between two circles should be at least 1/10 of the screen height. This is to prevent multiple detections of the same circle. 'param1' represents the threshold for the Canny edge detector used before the Hough Transform. A higher value for param1 means that only prominent edges will be detected, potentially missing some circles. Conversely, a lower value might result in detecting many spurious points. 'param2' is the accumulator threshold, which determines how many votes a set of parameters needs before I accept it as a circle. Similarly, if this value is set too high, some genuine circles may not meet the requirement and be excluded. 'minradius' and 'maxradius' define the acceptable minimum and maximum radii of the circles.

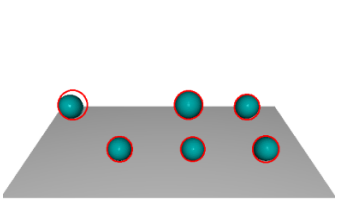

Figure 1: View0 Hough

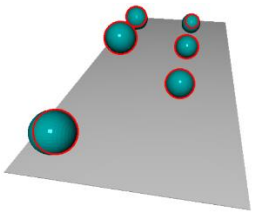

Figure 2: View1 Hough

2.2 Epipolar line

I have now obtained the expressions for all the circles in view0 and view1. Based on the epipolar constraint $P_R{}^T RSP_L = 0$, we know that for a point in view0, there is a corresponding epipolar line in view1. My next step is to find the epipolar lines in view1 that correspond to the centers of the circles in view0. According to the formula for epipolar constraints, we first need to compute the essential matrix, which is the product of R and S, where R represents the rotation needed to go from the coordinate system of camera 0 to that of camera 1, and S represents the skew-symmetric matrix of the translation vector. To convert the physical positions relative to the camera (described by E) into specific image coordinate relationships, we use the camera's intrinsic parameters and the formula: $F = M_R{}^T E M_L$, allowing us to transform the essential matrix into the Fundamental matrix. Additionally, we have the formula $p_R{}^T F p_L = 0$, which gives us the epipolar constraint in image coordinates. Thus, for each given circle center P_L in view0, we can find its corresponding epipolar line in view1! We iterate through all the circle centers in view0, identifying their corresponding epipolar lines in view1. Similarly, we can find the corresponding epipolar lines in view0 for the centers in view1.

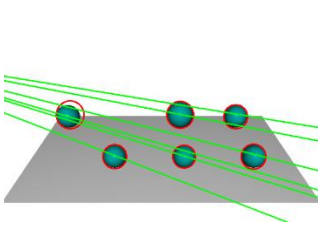

Figure 3: View0 Epipolar Line

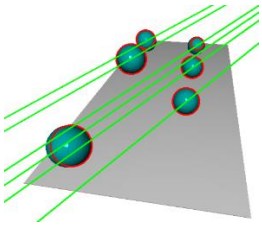

Figure 4: View1 Epipolar Line

2.3 Center Corresponding

Previously, for each circle center in view0, we identified its corresponding epipolar line in view1. Now, we need to precisely determine which point on this epipolar line corresponds to the circle center. My method involves iterating through each epipolar line in view1. For each line, I find the circle center in view1 that is closest to it in terms of distance. This circle center with the shortest distance is then considered the match for the circle center in view0. Additionally, to enhance the accuracy of the algorithm, I implemented a bidirectional matching approach. For every previous found circle center in view1, we reverse the process to find its epipolar line in view0 and use the same method to identify the corresponding circle center in view0. We compare this center with the initial one to check if they are the same. If they match, we save this pair of matching centers. If they do not match, indicating a failure in bidirectional matching, we discard these centers and do not include them in the reconstruction.

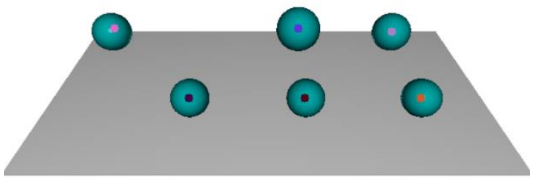

Figure 5: View0 Center Corresponding

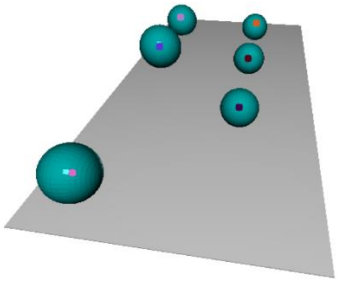

Figure 6: View1 Center Corresponding

2.4 Center Reconstruction

We have now established the matching pairs of circle centers between view0 and view1. Using these matched pairs, we can reconstruct the positions of the sphere centers in 3D space. First, we iterate through each pair of matched centers. For each pair, we calculate their respective coordinates in the camera's coordinate system. Then, using the formula1 and formula2, we compute H, which allows us to determine the parameters a, b and c. Once we have a, b and c, we can use the formula3 to calculate the coordinates of the reconstructed sphere centers in 3D space.

$$a\mathbf{p}_L - bR^T\mathbf{p}_R - \mathbf{T} - c(\mathbf{p}_L \otimes R^T\mathbf{p}_R) = 0$$

Formula 1

$$H\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{T}$$

Formula 2

$$\hat{\mathbf{P}} = (a\,\mathbf{p}_L + b\,R^T\mathbf{p}_R + \mathbf{T})/2$$

Formula3

2.5 Center Display

In this task, to facilitate the observation of the errors between the reconstructed sphere centers and the ground truth centers, I recreated a same empty plane to the one used earlier. However, this plane only contains the ground truth centers which are marked in red. I then recorded the results from the perspectives of both cameras, allowing me to visualize the ground truth centers on the camera plane images. Next, I iterated through the 3D sphere centers. For each center, I calculated its projection from the perspectives of the two cameras and marked it in green.

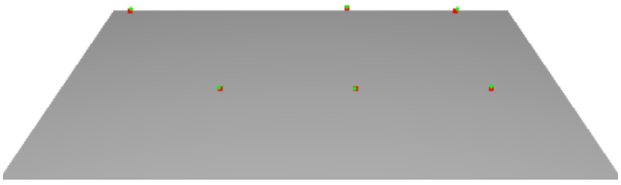

Figure 7: View0 Reconstruction center and Ground truth

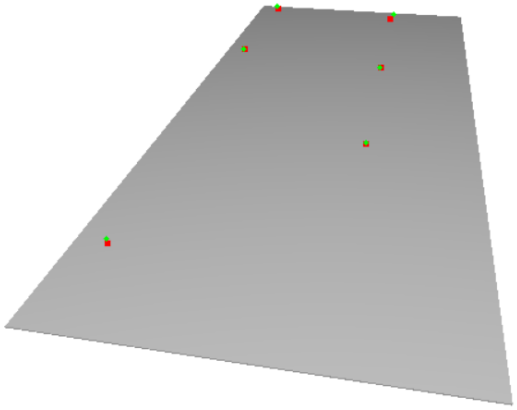

Figure 8: View1 Reconstruction center and Ground truth

2.6 Radius Corresponding

Now that we have reconstructed the sphere centers, to reconstruct the entire sphere, we also need to determine the radius of each sphere. My method is as follows: For each circle in view0, I select a point on its edge. For each of these edge points, I find the corresponding epipolar line in view1. The next step is to find the corresponding points. For each epipolar line in view1, we check if it intersects with the circle (this circle is the one to which the point in view0 belongs; since we have matched the centers, we can directly match it to the corresponding circle). If there is one intersection point, we make a direct match. If there are two intersection points, we choose the most suitable one. If there are no intersection points, we select the point on the circle that is closest to the epipolar line.

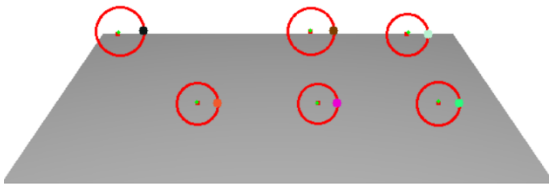

Figure 9: View0 Points Corresponding

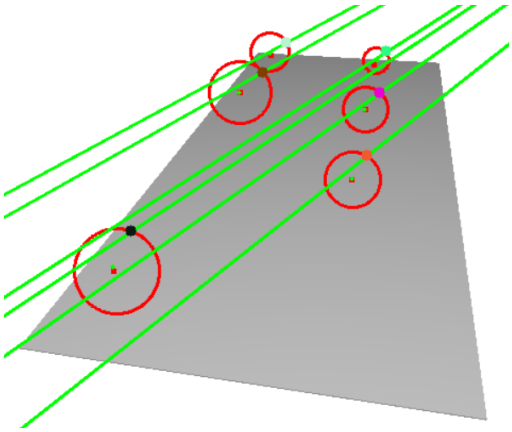

Figure 10: View1 Points Corresponding

2.7 Radius Reconstruction and Display

For each point on the edge of a circle in view0, there is a corresponding point in view1. Using the same method as in **2.4**, we reconstruct the position of these matched points in 3D. Then, using these points and the previously determined 3D sphere centers, we can calculate the radius of each 3D sphere. Now that we have the radii and centers of the spheres reconstructed in 3D, and considering that the spheres are solid, projecting both the reconstructed spheres and the ground truth together would not be conducive to clear observation. Therefore, I performed point sampling on the surfaces of both the reconstructed spheres and the ground truth spheres. Then, I re-projected these sampled points back into the views of cameras view0 and view1. This approach allows for a very clear observation of the discrepancies between the reconstructed results and the ground truth.

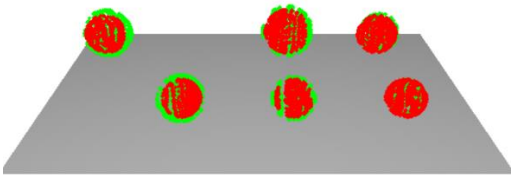

Figure 11: View0 Sphere Reconstruction

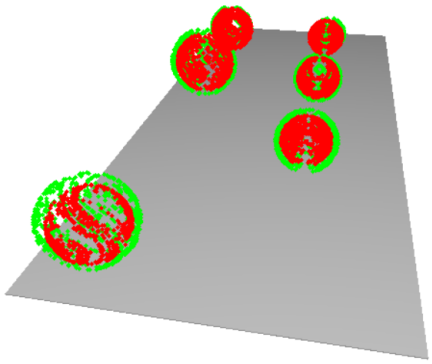

Figure 12: View1 Sphere Reconstruction

3. Experiments and Results Describe

3.1 Random Spheres

I made some adjustments to the part of the template that randomly generates spheres. I changed the default minimum and maximum spacing between the spheres, as the default values sometimes resulted in overlapping projections of two spheres on the camera plane. The overlap situation is showed in figure13 and figure14. Such overlap significantly impacts the results of the Hough Transform, so by modifying the default spacing, I almost entirely eliminated this issue. This was

crucial because if two sphere projections overlapped significantly, the Hough Transform might fail to detect a circle, hindering subsequent reconstruction.

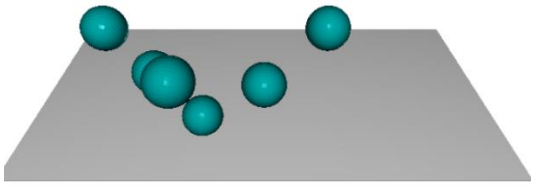

Figure 13: View0 overlap

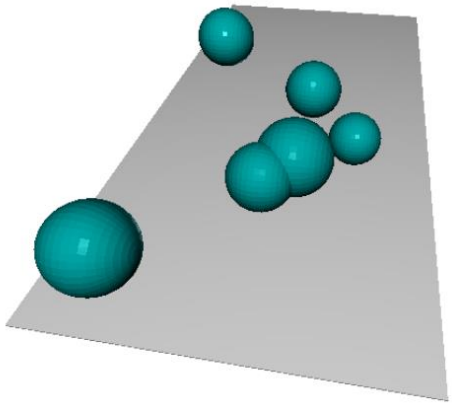

Figure 14: View1 overlap

3.2 Adjusting Hough Transform Parameters

During the Hough Transform, I tried numerous parameter combinations until I achieved satisfactory results. The Hough Transform needed to accurately detect all circles in both view0 and view1 without any false positives. I aimed for both TPR and F1 score to be 1 because a difference in the number of circles between view0 and view1 would lead to incorrect matching of centers, false positives could result in reconstructing non-existent spheres, and missing any circle would mean failing to reconstruct that sphere. Instances where TPR is not 1, as shown in Figure 15, indicate the presence of false positives, as illustrated in Figure 16. Thus, I adjusted the parameters to minimize these issues.

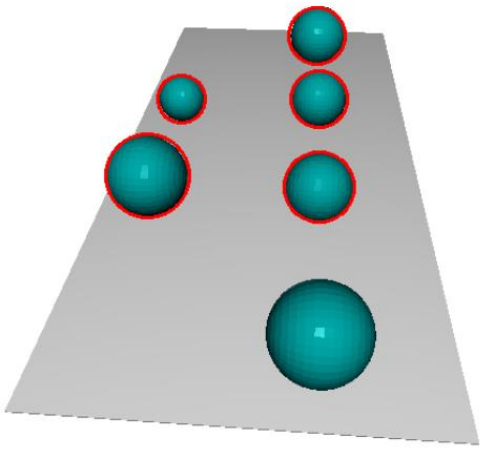

Figure 15: View1 TPR not 1

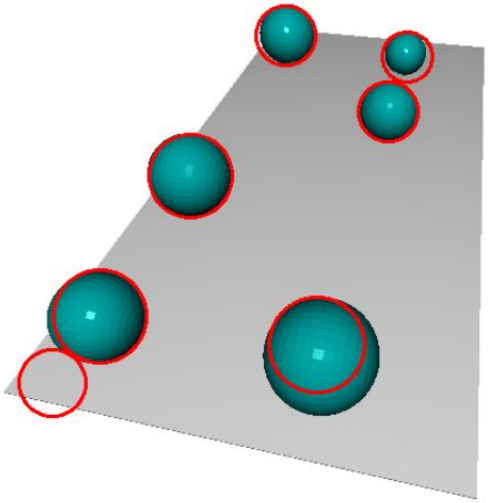

Figure 16 View1 FP not 0

3.3 Center Corresponding Error

In **2.3**, if bidirectional matching is not used for circle center matching, there is a high probability of mismatches. The case in Figure 17, other centers might be closer to the epipolar line than the actual center, leading to incorrect matches and subsequent erroneous sphere reconstruction. To avoid this, I used a bidirectional matching method, as explained in the method section, which significantly reduces the likelihood of mismatches.

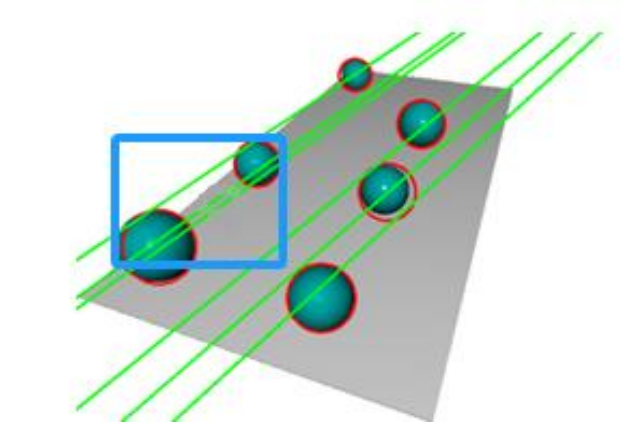

Figure 17: Mismatching centers

3.4 Radius Corresponding Error

In **2.6**, to correspond the radius of the spheres, I needed to find another set of matching points on each circle. Initially, I randomly chose a point on each circle in view0 and then found the corresponding point in view1. However, this led to mismatches showed in figure 18 and figure 19. And it will reconstruct the wrong 3D point and give the wrong radius, showed in figure 20. The method I employed was based on intersections, but a single epipolar line in view1 might intersect a circle at two points, making it unclear which was the correct match. To fix this, I observed the external parameters of the cameras and the function of the epipolar lines. I decided to select a specific point on the right side of each circle's center in view0, parallel to the x-axis. In view1, when an epipolar line intersected a circle at two points, I always chose the point with the larger x-coordinate, i.e., the point further to the right. This approach effectively solved the issue of mismatches at two intersection points. However, this method is effective when the changes in camera angles are not too big. It might fail if the random range of camera angles is significantly changed.

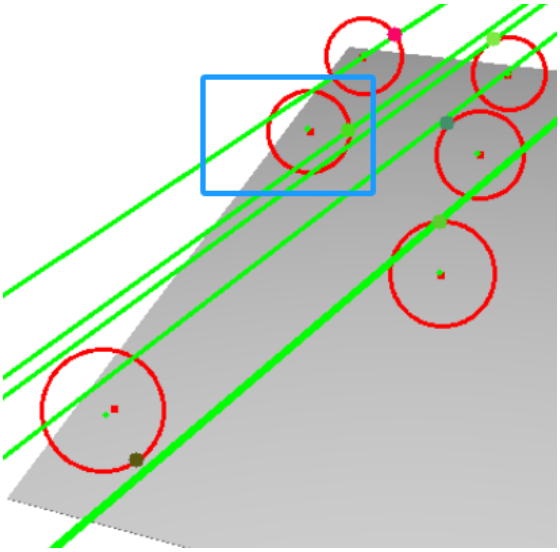

Figure 18: view1 Mismatching Points

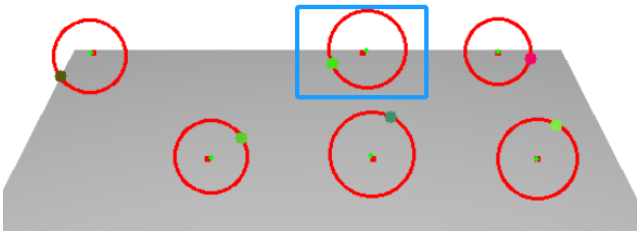

Figure 19: view0 Mismatching Points

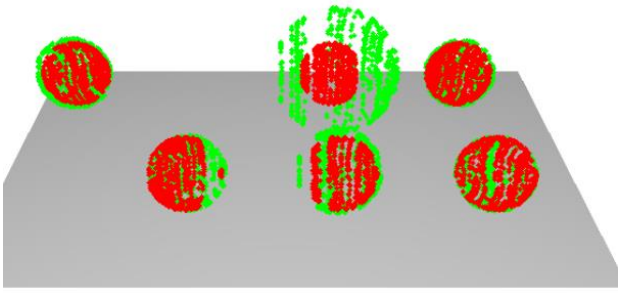

Figure 20: Wrong radius Reconstruction result

3.5 Radius Size Error

In the Figure21 and 22, the green spheres represent the 3D reconstruction results, and the red spheres are the ground truth. There are some small differences between the reconstructed results and the ground truth, mainly in the radii. These differences are due to inaccuracies in the Hough Transform, which may slightly overestimate or underestimate the size of the circles.

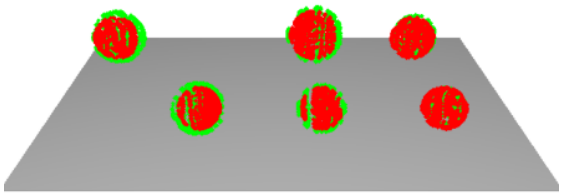

Figure 21: view0 Reconstruction Result

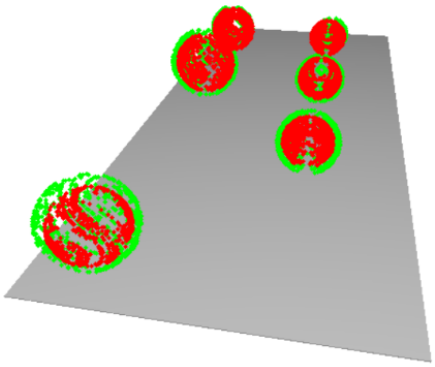

Figure 22: view1 Reconstruction Result

3.6 Camera Noise

Having completed the 3D reconstruction of the spheres, I proceeded to further investigate the impact of camera noise. For this purpose, I developed a function to add some noise to the extrinsic parameters of the two cameras. I then repeated the all of methods using these noise-augmented cameras to observe the differences between the reconstructed spheres and the ground truth.

Pictures on the left are the results without camera noise

Pictures on the right are the results with camera noise

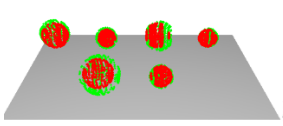

Figure 23: view0

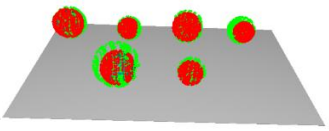

Figure 24: view0 with camera noise

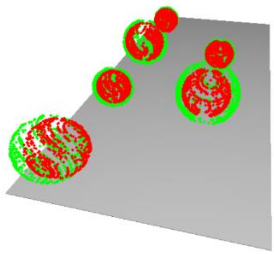

Figure 25: view1

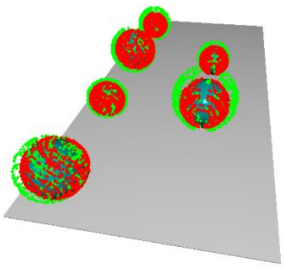

Figure 26: view1 with camera noise

From these images, it is evident that my algorithm demonstrates significant robustness to minor noise. Even with the introduction of noise, the differences between the reconstructed spheres and the ground truth remains minimal. This indicates that the algorithm is reliable in handling slight inaccuracies that might be encountered in practical applications.

4. Conclusion and Further improvements

In this experiment, I successfully completed the reconstruction of a 3D object using methods epipolar constraint, point matching, and triangulation reconstruction. The final results demonstrate that the reconstruction is quite effective. However, due to the precision issues with the Hough Transform, there may be some errors in the final results, but these are within acceptable limits. In the future, I plan to further enhance the accuracy of the Hough Transform to improve the effectiveness of 3D reconstruction. Additionally, as the camera angles are randomly set within a certain range, the method I used for matching points on the sphere radius may only be suitable for specific cases and lacks universality. Moving forward, I intend to use more complex methods for point matching, such as SIFT, to address this limitation.