18/12/2018

RALAZABA ELECTRONICS

Weekly Report

Team Members: Ali AYDIN, Anıl AYDIN, Enes AYAZ, Nail TOSUN, Selman DİNÇ

Advisor: Lale ALATAN

Done

- We tried to control the mouse on MATLAB and we drew the path of mouse without using any rotation angle. We just drew on XY plane. The result can be seen in Figure 15.
- We tried to obtain rotation angle by using gyroscope, we realized it only gives information about the direction gravity vector and its rotational output is not reliable. Due to that, we decided to use compass or two mousses to sense rotation of robot.
- We created a simulation environment for shape and center detection. The maps and shapes are in the section of “Shape and center detection simulation”
- After that, we tried to use measured data for shape detection, but our measured data needs to be cleaned in some aspect. For that, we made a modification in our algorithm. The explanation of the modification and the results with measured data is in the section of “shape finder algorithm for measured data”.

To Do

- We will start to write conceptual design report on Wednesday.
- We will try to obtain rotation angle by using two mouse or a mouse and compass.

18/12/2018

Shape and center detection simulation

To test our first algorithm for shape and their center detection, we created simple maps with no error using paint. The maps that we tested are shown in from Figure 1 to Figure 4.

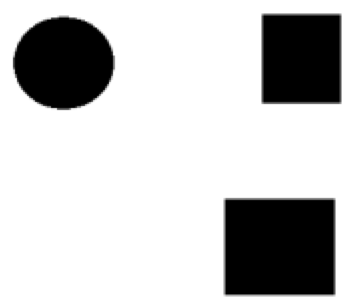


Figure 1: First Test Map

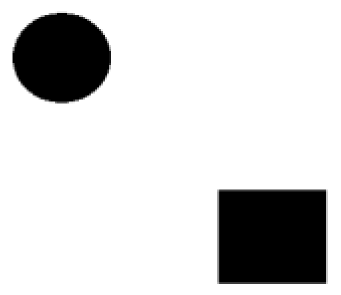


Figure 2: Second Test Map

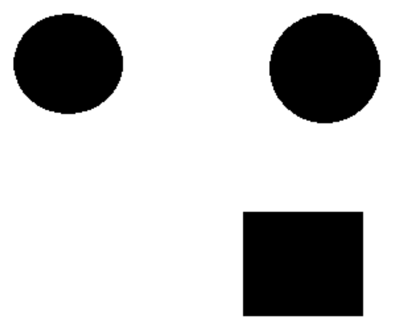


Figure 3: 3rd Test Map

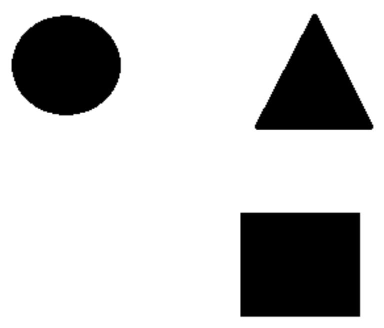


Figure 4:4th Test Map

We created a simulation environment such that, we loaded these maps to a MATLAB script and we selected 3 points for sensing these shapes. From these 3 points, we created lines with different slopes and the intersection points represented the lidar scans. These lines will represent how the map will be seen by the lidar sensor. Our simulation environment is seen like in Figure 5. In the figure, the lines (lidar measurements) is intercepted with the boundaries of the objects. The interceptions are shown in the figure as red dots. The intersection points for 4 different map configurations are shown in from Figure 6 to Figure 9.

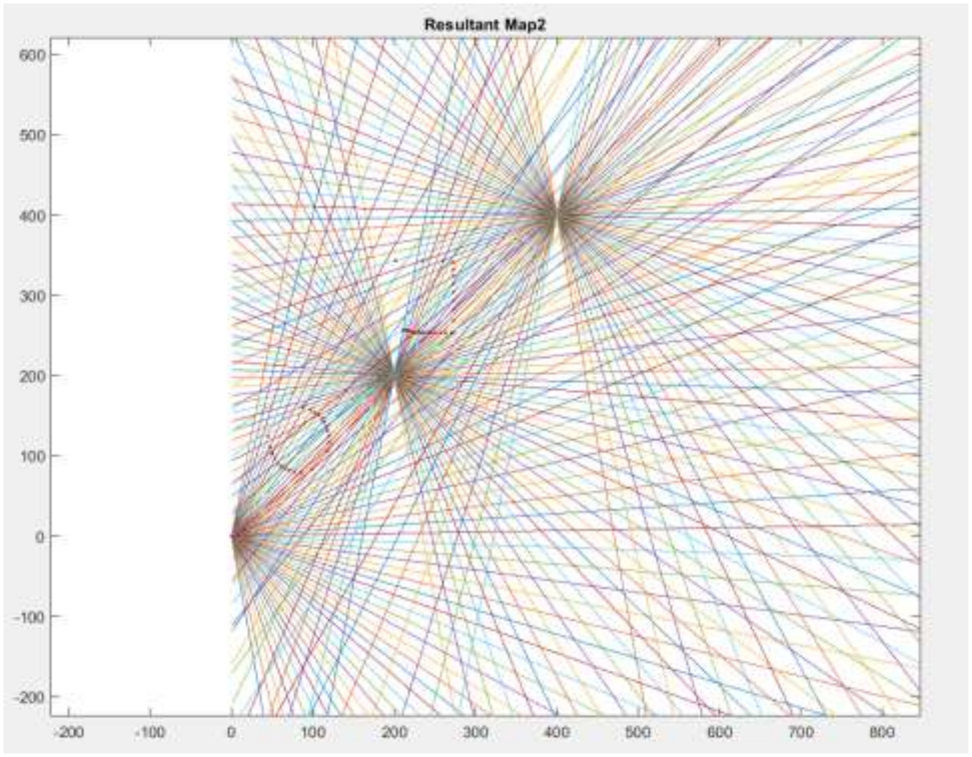
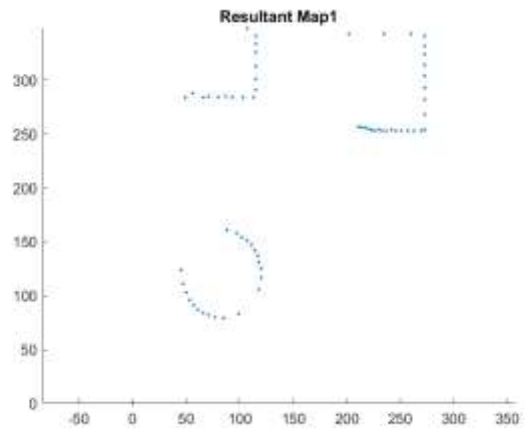


Figure 5: The simulation environment

After intersecting lines and our objects, some parts of objects are not shown which will be the case in real data. The missing parts of the objects can be seen in resultant maps. We must come up with this problem first. To do that, we combined x and y coordinates of resultant map data and converted these data to an image to be able to process and create objects.



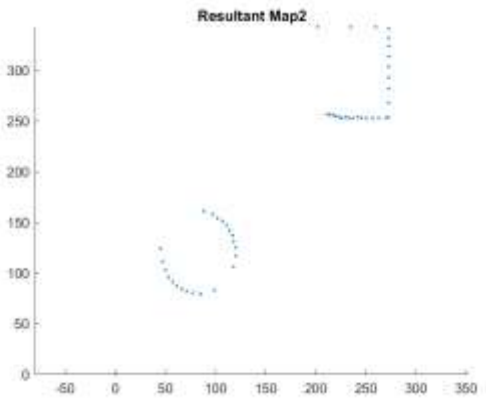


Figure 6: Lidar Scan Result for 1st Map

Figure 7: Lidar scan result for 2nd Map

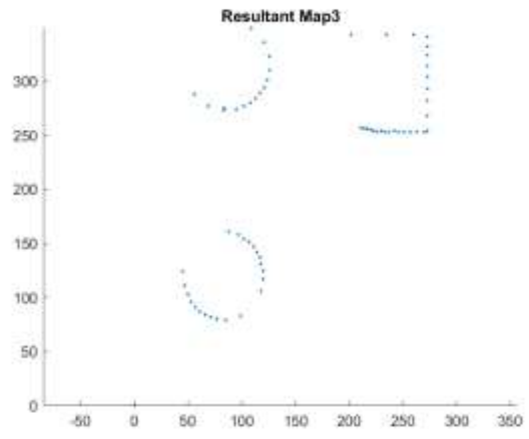


Figure 8: Lidar Scan result for 3rd Map

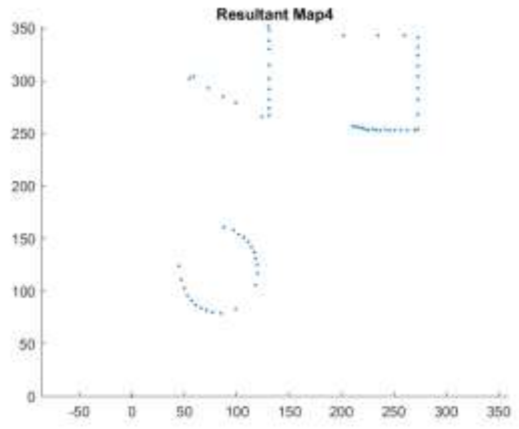


Figure 9: Lidar Scan Result for 4th Map

Thus, after simulation, we converted resultant maps to images and made threshold operations. The thresholding operation is used to determine the boundaries and edges of the objects for Multiple Kernels. Further readings can be found under the topic "Multiple Kernels for Object Detection". After thresholding, we found contours in the image and using some constraints such as the area of contours and the length of contour arrays, we created 3 different cases for rectangular, circular and triangular objects. The resultant maps from the missing data are shown in from Figure 10 to 13.

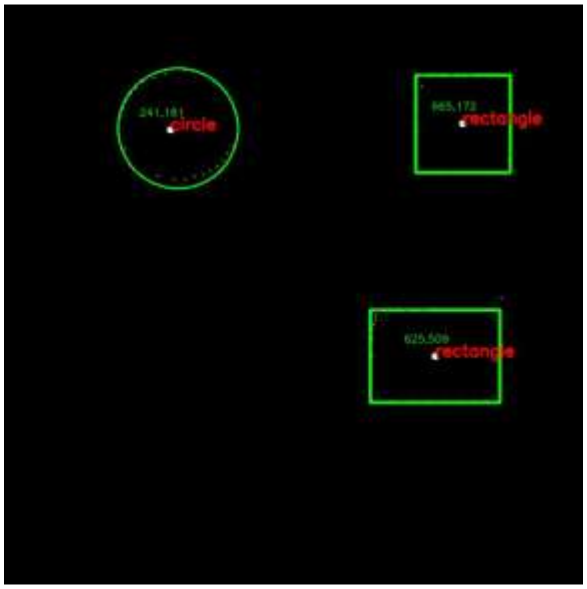


Figure 10:Resultant shapes and centers for 1st Map

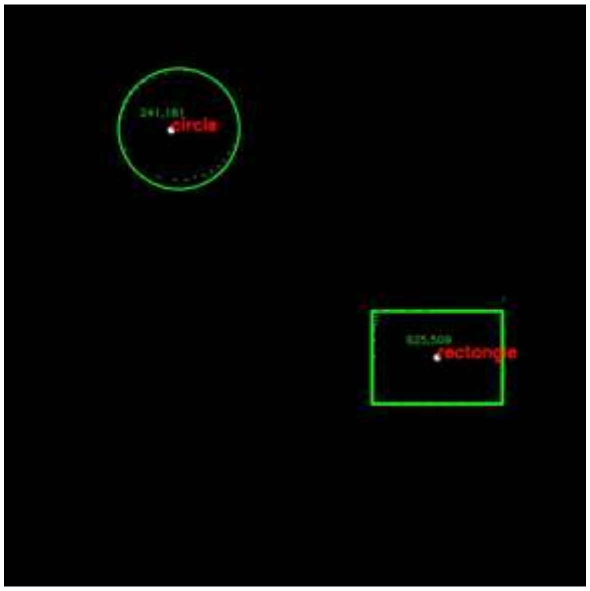


Figure 11: Resultant shapes and centers for 2nd Map

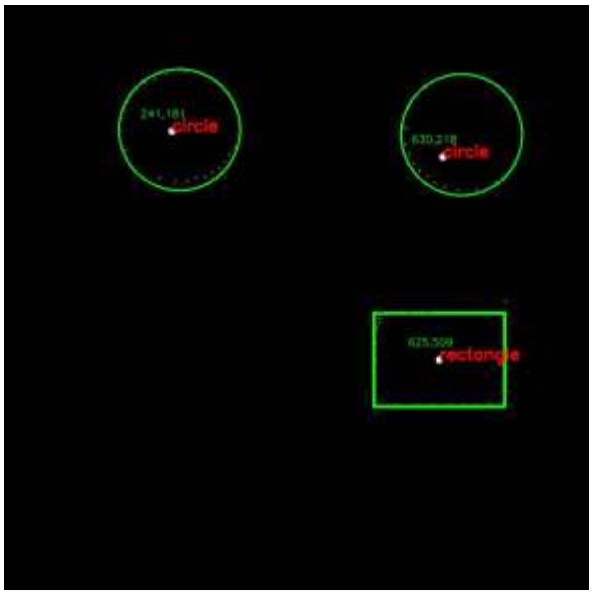


Figure 12:Resultant shapes and centers for 3rd Map

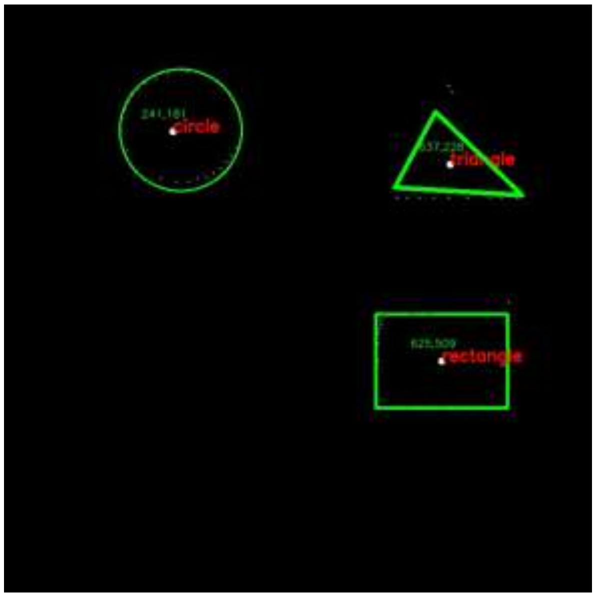


Figure 13:Resultant shapes and centers for 4th Map

Yet, we have some problems about our algorithm which is detection of the triangular object is not working well. We will solve this problem by defining the triangle dimensions to the software as we know them.

Measurement

We set the measurement environment to test our mapping algorithm with real data. The setup, Figure 14-16, has 3 different viewpoints with respect to object. The test objects are scanned via the lidar sensor from three different points. These measurements are merged by using an algorithm that takes a position of sensor (in 2 Dimension) and angle between stationary coordinate system and sensor data. Normally, the lidar will measure the distance with related angle. This angle information is in the frame of the lidar. We need to project the measurements to the global frame of the map. To accomplish this task, the measurements are rotated using rotational transformation matrix and positioned according to the position vector of the lidar. The output of algorithm is illustrated in Figure 14 to 16.

Later, the obtained data will be fed to the object finder algorithm.

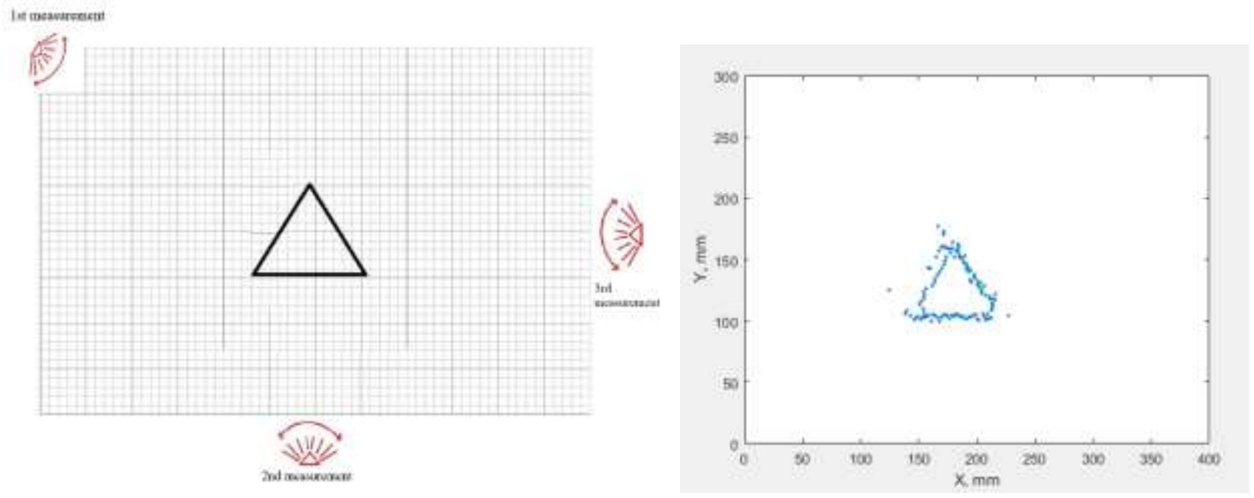


Figure 14 First Test Map and its output

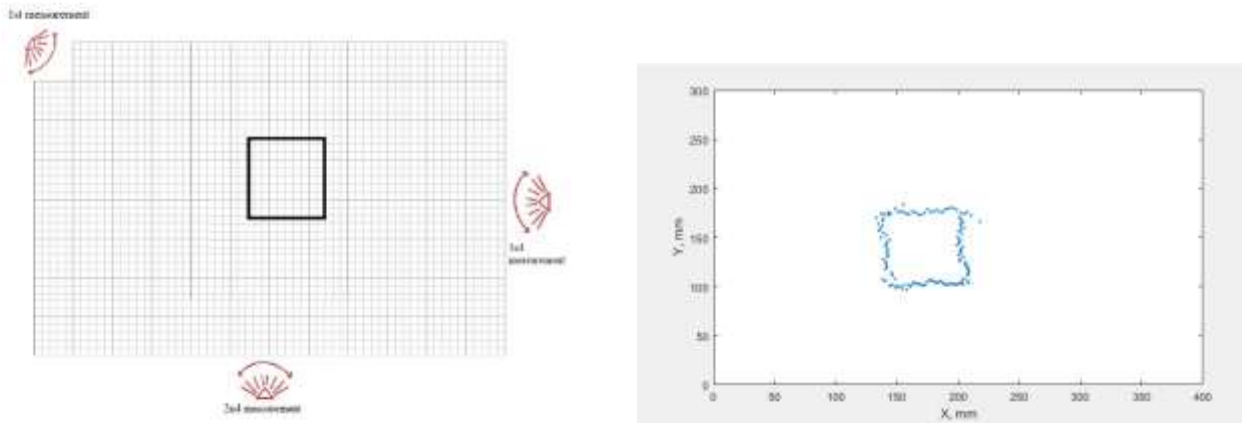


Figure 15 Second Test Map and its output

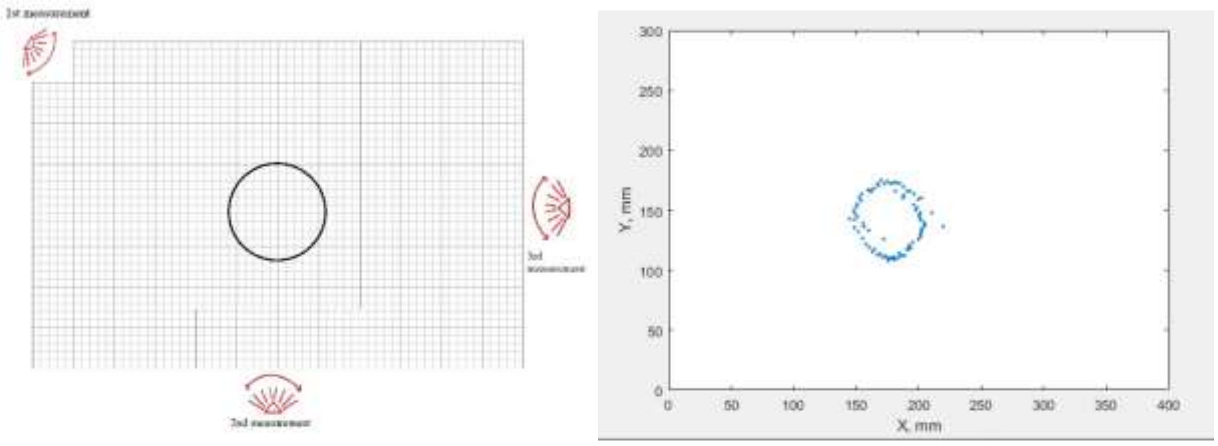


Figure 16 Third Test Map and its output

Outputs of shape finder algorithm for real measured data

When we tested our shape finder algorithm with real erroneous data, we had some problem. Most important problem is that the spaces between measured points are large and this causes that our algorithm can find wrong contours. To eliminate this, we added new points to resultant array by adding neighbor points of real data clouds. After that, our thresholding operation worked and we were able to find true contours. The output for rectangular object is shown in Figure 17. We found the center of rectangle as 146,135. The real center was 146,140. This means that we had 3% error for y coordinate of center.

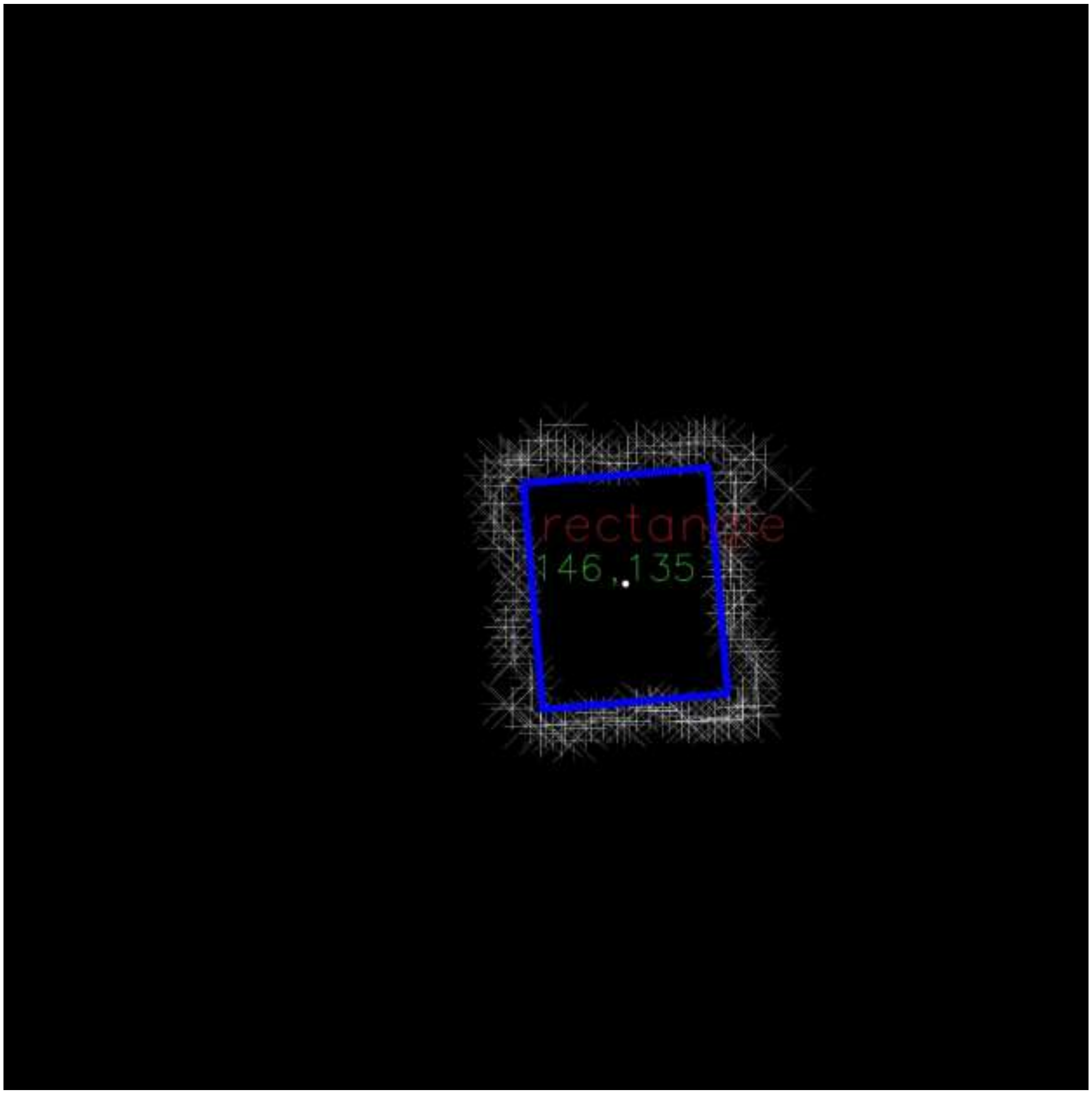


Figure 17: Output of rectangular object

The output for cylindrical object is shown in Figure 18. For cylindrical object we found the center as 154,135. Real center was 146,140. Thus, we had 6% error for x coordinate and 3% for y coordinate.

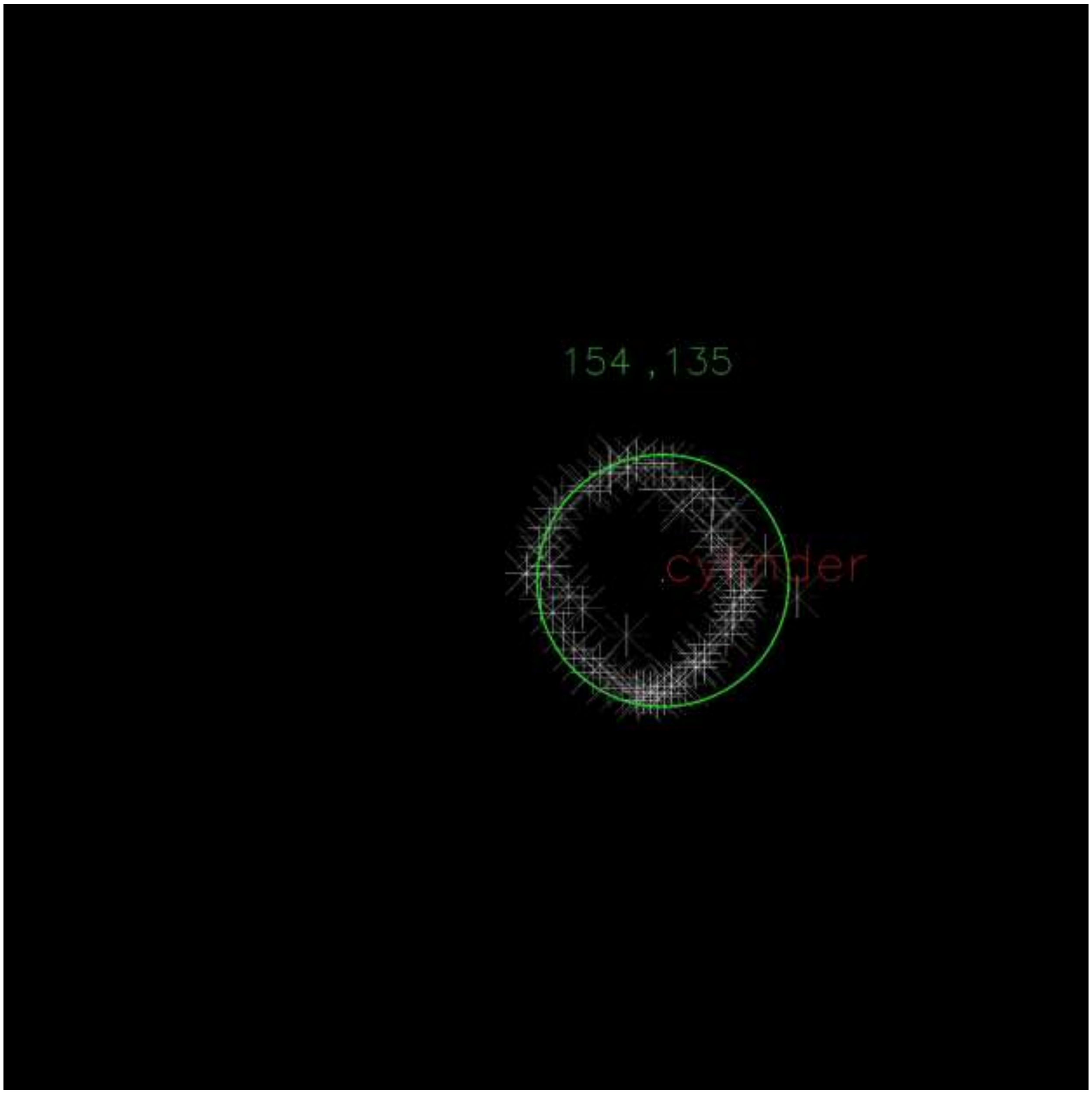


Figure 18: Output of cylindrical object

The output for triangular object is shown in Figure 19. Here, we found the center as 148,146. Real center was 184,130. We had 24% error for x coordinate and 12% error for y coordinate.

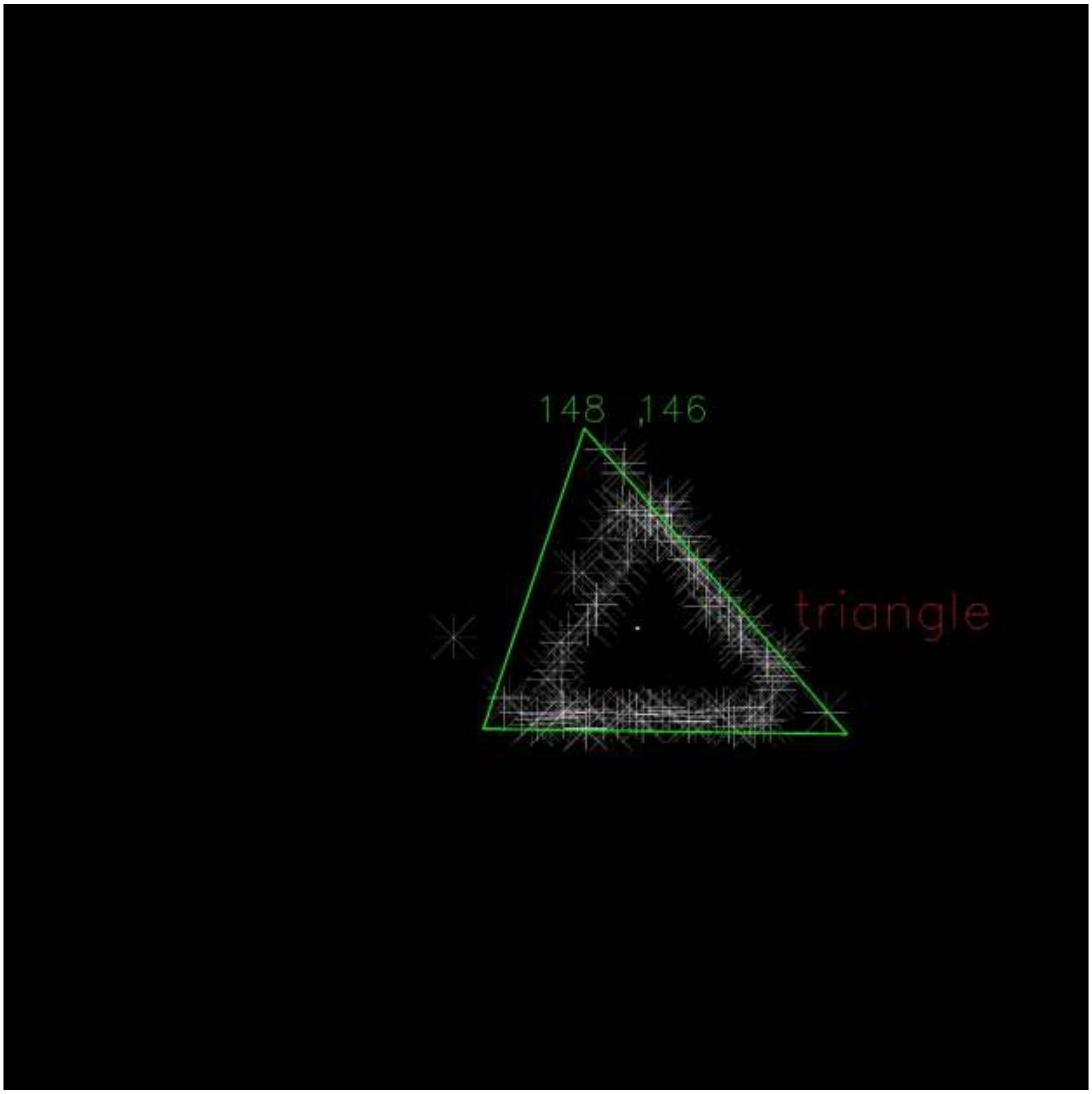


Figure 19: Output of triangular object

These results show us that, while measuring data, we need to have some improvements for better mapping since we need very clear input for shape finding in our case. Another result is that to detect and find the shape of triangular object is hard and, we must modify our shape finding algorithm for more accurate results.

For Mouse Application

We connected a trackball mouse to Arduino and to obtain a graph, and we took data on serial port by using MATLAB. Therefore, as seen Figure 20, we track mouse on the graph. To obtain tracking line, we drew vector between two point that sensed from mouse in a certain time. The resolution of tracking is dependent to the DPI (Dot Per Inch) ratio. In our case, we used 255 Dpi mouse, then we has 100 micrometer resolution of tracking.

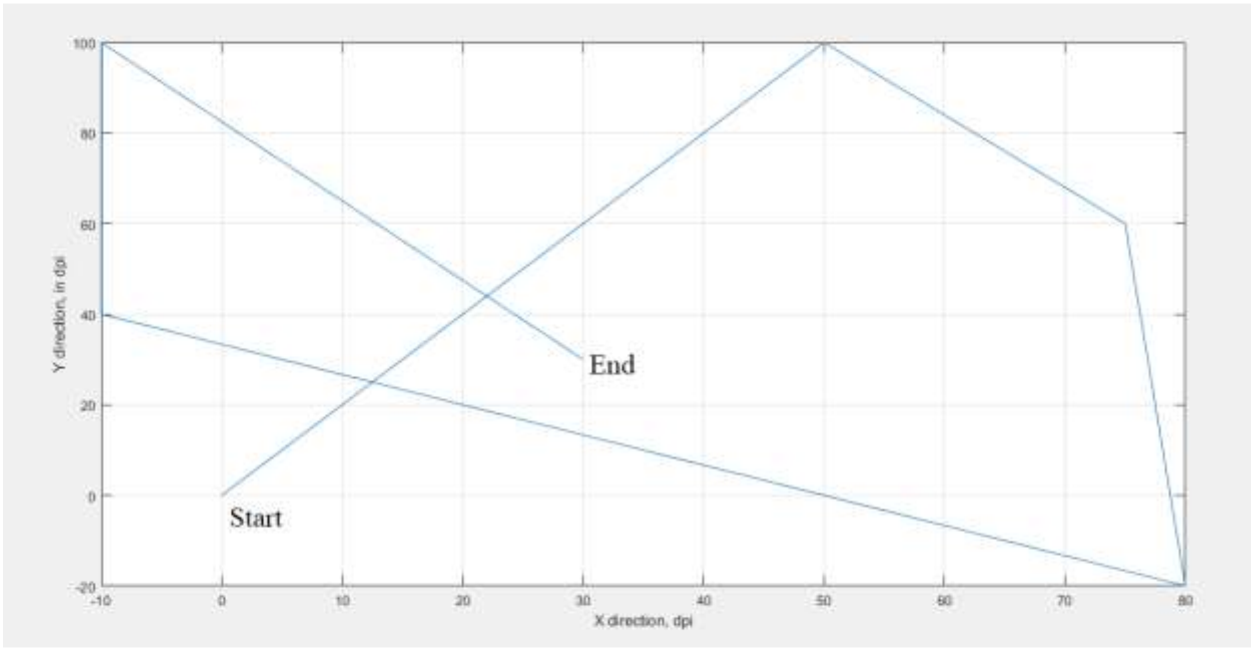


Figure 20: The path realization for arbitrarily-moving Trackball Mouse.