Terrance Williams

July 6, 2023

1 Introduction

Project 2 is a continuation of the work done in Project 1. In the previous project, a basic tennis ball detection algorithm was used to determine the presence of tennis balls in a given camera frame. The detection algorithm uses a combination of color filtering and Hough Circle detection—which detects circle centers—to look for the characteristics of a tennis ball, namely its yellow color and spherical shape. As a result of the implementation method, uncertainty in the results are introduced to the system in the form of noise, making it more difficult to determine what was a true detection and what was a false positive.

This project aims to correct this behavior, fine-tuning the detection method and creating a way to differentiate the real tennis balls from the noise. Finally, once a real tennis ball is determined, the angle between the robot and the ball will be calculated and used to perform corrective rotation, resulting in the robot facing the ball directly.

2 Handling Noise

Phase One of the project was used to reduce the effects of noise on the system, beginning with minimizing its general presence. Throughout this project, initial tests were done in a testing environment[1] to then be incorporated into the robot after the concepts were proven to work. This section will discuss the results of the testing environment in order to separate the general concepts from the ROS-specific implementation details (which add more complexity).

2.1 Parameter Tuning

To reduce the total amount of noise in the system, I performed real-time parameter tuning. OpenCV provides the ability to create trackbars for various parameters of the image, and coupled with video streaming, one can get live feedback

[1] Testing environment included my laptop and office space.

of how a given variable setting affects the performance of the algorithm, [CITATION NOTE][CITATION NOTE]. I tested four 'categories' of control variables: color masking (HSV values), Hough Circle parameters, image contrast, and image smoothing. Minor tweaks were made to the acceptable range of HSV values as well as the Hough parameters and contrast, however the greatest positive effect on noise reduction was how the image was smoothed. In Project 1, captured images were only smoothed one time by using a median blur filter. What I found through tuning was that a combination filter has better performance, especially when done multiple times per image. The presence of noise in the testing environment was reduced considerably by passing each image through a median+Gaussian filter three times. The result was a much smoother image that more accurately detected round shapes. Of course, the specific parameter values change when using the robot, but the general concept is the same.

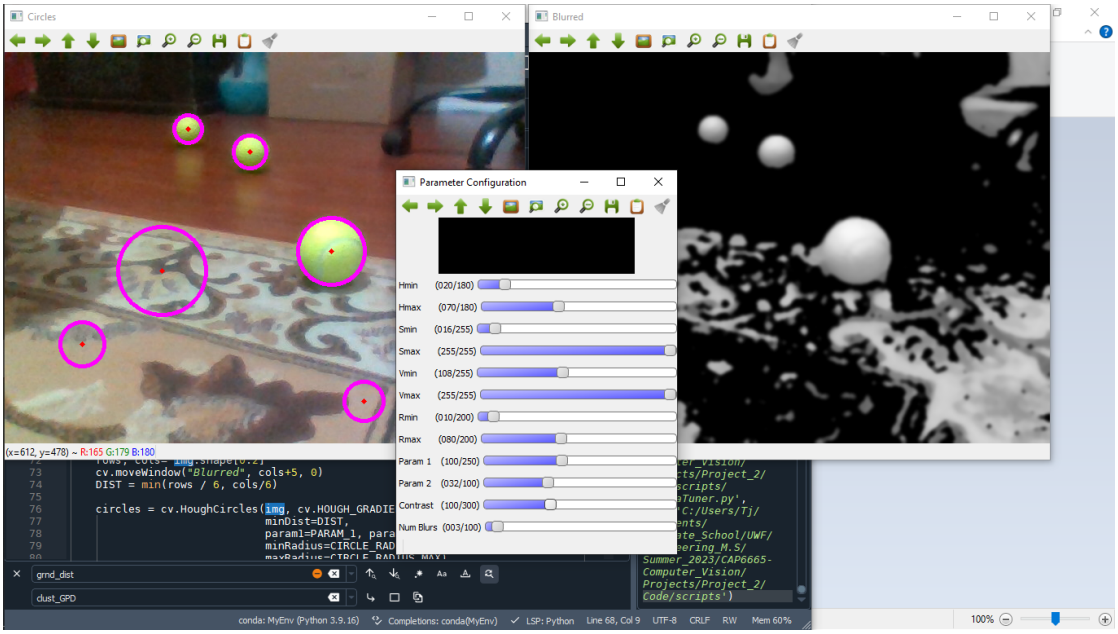


Figure 1: Trackbar Tuning Example (Test Environment)

2.2 k-Means Clustering

After adjusting the parameter values, the next step was to separate the remaining noise from the desired data. As seen in Figure 1, even with parameter tuning, undesired detections are still present in the system. To separate these outlier points from the tennis ball data, I used k-Means clustering.

2.2.1 Justification

The data's characteristics were the primary driving factor for choosing k-means clustering as the project's data-filtering method. Because the robot is meant to collect tennis balls, it can be assumed that the tennis balls of interest will be stationary. Since the robot will also remain motionless when scanning for balls, this means that Hough-detected circle centers corresponding to tennis balls should be positioned close together over the course of the detection period. Noise, however, is much more randomly assorted. As a result, if the data can be segmented into clusters, the clusters with small pixel area (high density) are more likely to be tennis balls.

2.2.2 Determining the Amount of Clusters

In order for k-means to perform properly, however, the correct k value—the number of clusters—must be chosen according to the data. Since the presence of system noise is dynamic, the choice of k must be dynamic as well. Otherwise, if there is a detection period with very little noise for example, a constant k-value may result in multiple clusters whose points all belong to one tennis ball, effectively segmenting one tennis ball into multiple.

Recall from Project 1 that the ball detection algorithm requires a number of consecutive detections, α, for a detection to officially occur. This α is the detection period (ex. 75 images). Throughout this period, there will be a frame that has the largest number of individual circle center detections. I use this image to determine the k value for a given clustering operation.

For example, imagine the environment has only one tennis ball. Ideally, all α images throughout the detection period would have only one detected circle center per image. However, due to the presence of noise, a given image may have multiple detected circle centers. Now imagine that over this detection period, the maximum number of individual detections was three. This means that at some point in the detection period there was an image with one detection that was legitimate and two that were noise. Therefore, we know there must be (at least) three clusters,

two for noise and one for the ball[2].

Figure 2 provides a visual example of the result of this operation. The left image is the final image of the detection period (the αth image). There are five detections in this image: one detection is the tennis ball and the other four, noise. The image on the right is the graph-representation of the k-means operation. Note that there are seven clusters. This means that at some point in the detection period there was an image with seven detections. Also notice that the cluster belonging to the tennis ball, Cluster 4, is much more densely-concentrated than the other clusters, lending credibility to the method used to distinguish legitimate data from noise.

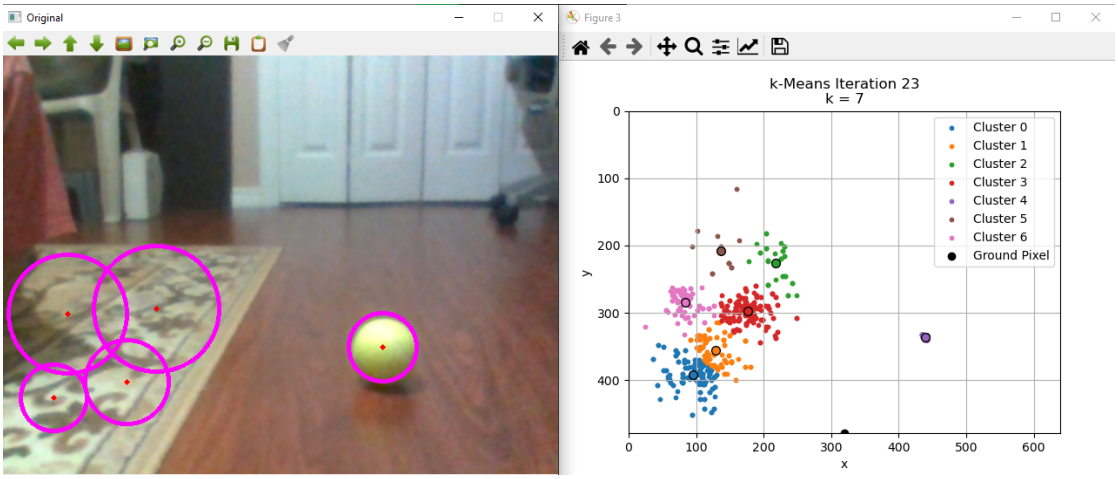


Figure 2: K-Means Clustering of the Detection Period

2.2.3 Setting Initial Means

Instead of selecting the points at random, I decided to choose the initial means for clustering because of the possibility that multiple points from the same tennis ball could be chosen as separate mean, dividing the tennis ball into two or more clusters.

Fortunately, the same image that determines the k value determines the initial means. For each detected circle on that image, the coordinate of the circle center is used as an initial mean. So, if we have a k of three, there are three matching initial mean points. Setting the means this way ensures that the noise points in the max detection image each belong to their own cluster.

[2]While noise points appear random from the perspective of the human observer, they are not from the perspective of the computer. If the computer detects noise in some location, other noise points may be in the same general area, hence including the noise point in the cluster count. Even if only one noise point appears in the whole period, we still want to isolate it from the legitimate points.

2.3 Choosing the Correct Cluster

Along with being able to cluster the data, the program needs to be able to select the correct cluster. To do so, it considers three characteristics: the number of points the clusters have, point concentration (density), and the vertical distance of a given cluster from the bottom-center pixel (which I will refer to as the ground-pixel distance, GPD).

2.3.1 Metrics

The number of points a cluster has is used as a metric mostly because density is used as a metric. Consider what happens if a noise cluster has only one or two points close together. The cluster would have a high (potentially infinite) density which would skew the results. Tennis ball clusters should have a high number of points along with a high density, so removing the clusters with a low number of points will prevent low-point, high-density issue.

The minimum number of points a cluster must have to be considered a 'candidate' is a proportion of the detection threshold, α. Ideally, a legitimate tennis ball cluster would have α points, but missed detections also happen on occasion, so instead a candidate must have $p\alpha$ points where $p = 0.8$ at the time of writing. So, if the detection threshold is 100 consecutive detections, a candidate cluster must have at least 80 points. Otherwise, the cluster is essentially discarded.

Density is used as a metric because tennis ball cluster points are extremely close in proximity, so each point's distance from the cluster centroid should be small. Therefore, clusters with high density are more likely to be legitimate tennis balls than clusters with lower densities.

Finally, ground-pixel distance (GPD) is used to help the case where there is more than one legitimate tennis ball present. In that case, a tennis ball collector would want to choose the closest ball to it, which means the balls closer to the bottom of the image. If two or more candidates have similar densities, GPD is a tie-breaker of sorts.

2.3.2 Density Calculation

The initial method used to calculate density was the blanket use of maximum intra-cluster distance. For a given candidate cluster (meets the minimum point count), the centroidal distance of every point in the cluster is calculated. The maximum distance is then chosen as a radius of a circle to serve as a cluster boundary. The area of this circle is then used in the density calculation:

$$\rho_i = \frac{n_i}{A_i} = \frac{n_i}{\pi r_i^2} = \frac{n_i}{(\text{max centroidal distance})_i^2}$$

where ρ_i, n_i, A_i are the density, point count, and area of the i^{th} candidate cluster.

The problem, I found, with this method is that is extremely sensitive to outliers. If a stray noise point happens to be placed in the tennis ball cluster, it ruins the density calculation, resulting in a density potentially multiple orders of magnitude smaller than it should be.

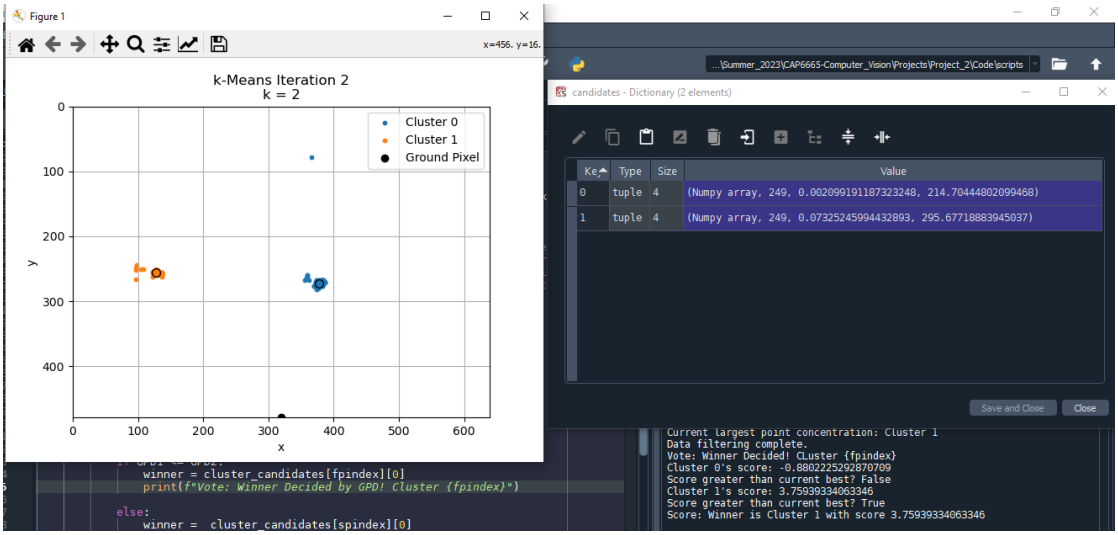


Figure 3: Effect of an Outlier on Cluster Density Calculation

Figure 3 shows the effect a single outlier has on the density calculation. Clusters 0 and 1 are both visually similar save an extreme outlier point for Cluster 0. In fact, without this point, Cluster 0 should be the winning cluster since it is slightly closer to the ground pixel. However, because of the outlier point, its density is calculated to be ≈ 0.0021 compared to Cluster 1's 0.073. With the outlier included, Cluster 1 becomes the clear winner, which is not what the program should decide.

To fix this behavior, I had to determine how humans are able to immediately see the outlier. Where is the boundary between inliers and outliers? Consider the following table of values:

Sample	y
0	2
1	4
2	8
3	14
4	22
5	87
6	95
7	101

Sample 5 hopefully attracted the reader's attention because the jump in value from Sample 4 to 5 is so much greater than the others. The same applies to the visual example. We can see the outlier so clearly because it is so much farther away from the centroid than the other points.

To find the inlier/outlier boundary, I find what is essentially how the data "accelerates," or how the change in data values changes. Going back to the table:

Sample	y	y'	y''
0	2	-	-
1	4	2	-
2	8	4	2
3	14	6	2
4	22	8	2
5	87	65	57
6	95	8	-57
7	101	6	-2

we can see that Sample 5, the first outlier point, has the greatest positive acceleration compared to the other points. Sample 5 is boundary between inliers and outliers of this set of data. I use this concept to find the boundary for the distance values of a given candidate cluster by doing the following:

1. Calculate centroidal distances for every point in a cluster.
2. Sort the distances from least to greatest to get a list of distances, d
3. Subtract every $d[n-1]$ from $d[n]$ to find velocity v, which represents how the data is changing w.r.t each sample.
4. Subtract every $v[n-1]$ from $v[n]$ to find acceleration a, which represents how the velocity is changing w.r.t each sample.

5. Find the maximum acceleration. If it is above a specified value, outliers exist in the data starting at boundary, Sample x. Choose instead the distance two samples away Sample $(x-2)$[3].

Plots of the above method are included in the Figures 4 and 5 for a visual aid. More can be found in the Appendix. It was observed that "good" clusters typically had points max accelerations of about two. To be safe, the threshold for outlier boundary is five. If a cluster has an acceleration above five, the outlier points are ignored for the choice of cluster radius.

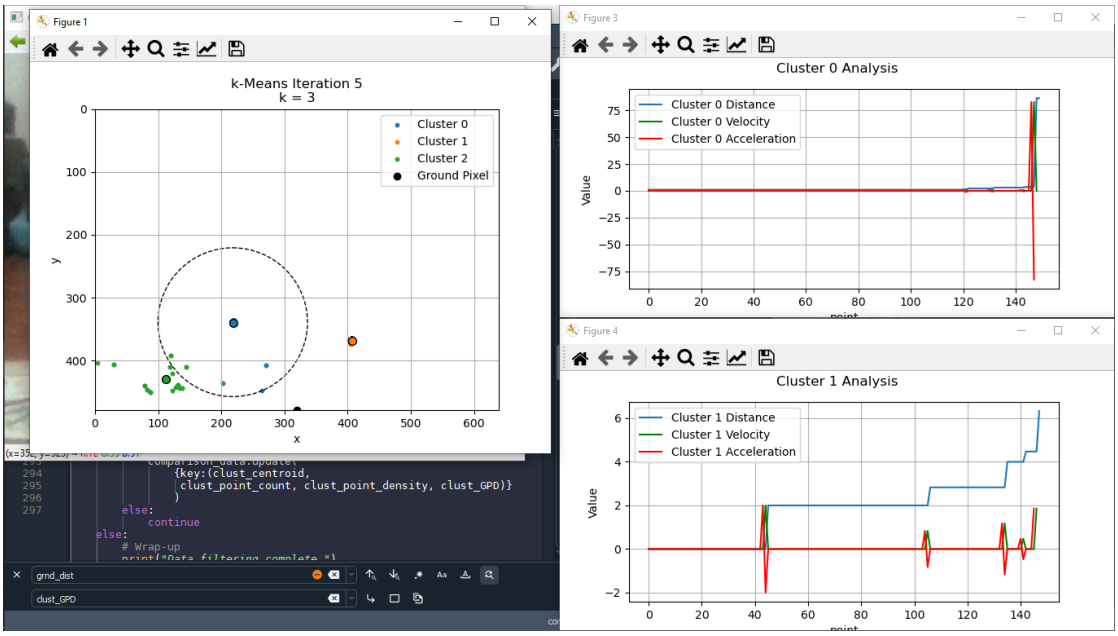
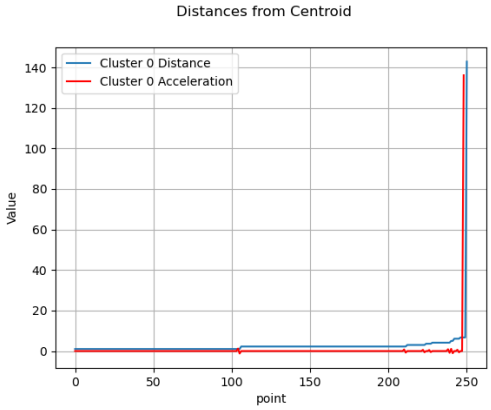
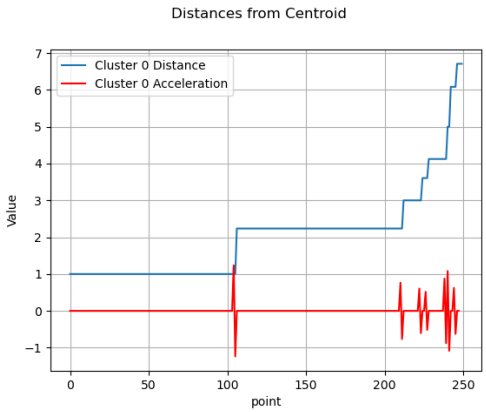


Figure 4: Acceleration Plot for two clusters. Cluster 1 had low acceleration while Cluster 0 peaked at 75.

[3]This is a result of how this method is implemented programmatically. If $max(a)$ has index j, the corresponding index in d is $j+2$: $a[j] \leftrightarrow d[j+2]$. Therefore, $d[j]$ is the distance value two samples away.



(a) Acceleration with Outlier



(b) Outlier Removed

Figure 5: Acceleration Plot of a Test Cluster with and without the outlier point.

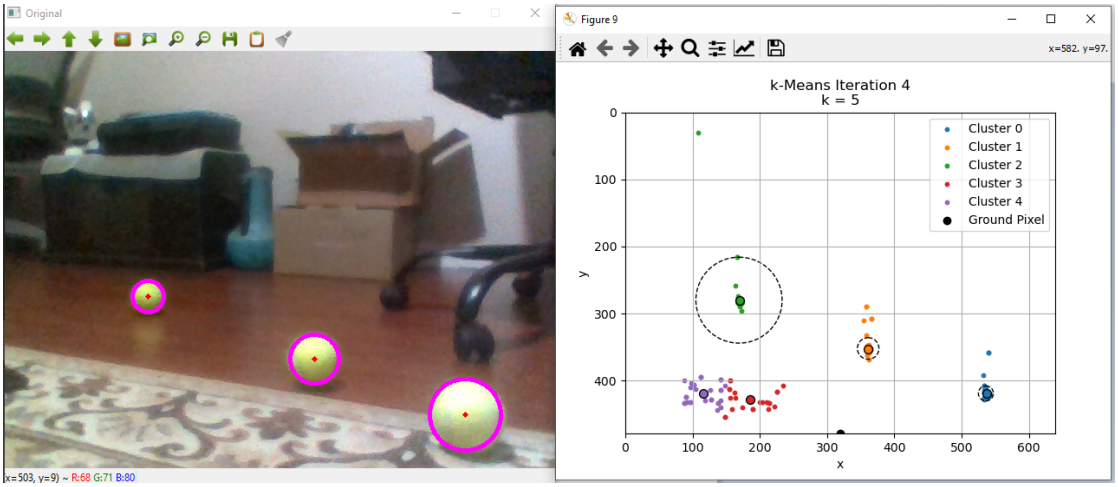
To test the method, I introduced acceleration into the system by tilting my laptop screen during a detection period (tests utilize the laptop's camera). the results are shown in Figure 6. The clusters with the highest accelerations had their outliers removed. This is especially observable in Cluster 2. Without this method, the enclosing circle would have been enormous.

3 Selecting the Winning Cluster

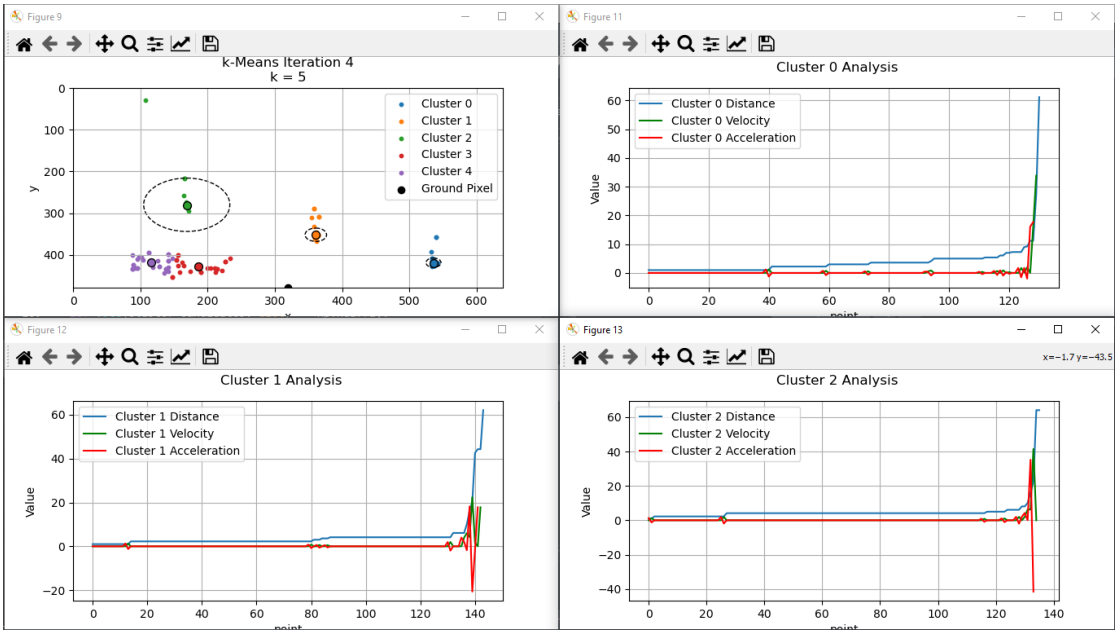
With the completion of the preceding steps, the detected ball data can be filtered and judged by the computer to decide on the 'winning' cluster. I approached the selection method two ways: scoring and voting. For scoring, I attempted to create a linear scoring function that gave weight to each of the three metrics, but the results were not satisfactory compared to the voting method due to the arbitrary nature of the scoring function. This may be a path worth pursuing again in the future, but for this project, the voting method was chosen.

The vote is a filtering process that proceeds in three stages. The first stage is the aforementioned minimum point count. Any cluster not meeting this count is disqualified. The remaining contenders move to density comparison. I iterate through the remaining candidates, tracking which clusters have the highest and second highest densities. These two clusters become the "first-place" and "second-place" clusters. Once these are found, I calculate a ratio between the clusters' densities:

$$R = \frac{\rho_{1st}}{\rho_{2nd}}$$



(a) Given Clustering Example



(b) Acceleration Plot

Figure 6: Effects of the Intra-cluster Outlier Removal Method

if $R \geq 1.75$ the current first place cluster is selected as the true winner because its density is notably higher (meaning it's more likely to be a tennis ball). If the ratio falls under this value, the vote proceeds to stage three.

The third stage is a GDP comparison. The cluster closer to the ground pixel is selected as the winner.

In the event two clusters have the exact same density and exact same GDP, the program selects the cluster it encountered first.

Upon selection of a winner, the voting function returns the centroid coordinates of the winning cluster for the adjustment angle calculation.

Figure 7 shows an example of the voting (and scoring) functions for two very similar candidates. The detection threshold for this test was $\alpha = 150$, so contenders needed $0.8\alpha = 120$ to enter the contest. The two contending clusters had very similar densities with a result ratio of $R = 1.076$). As a result, the winner was decided by GDP, with Cluster 0 being closer to the ground pixel than Cluster 1.

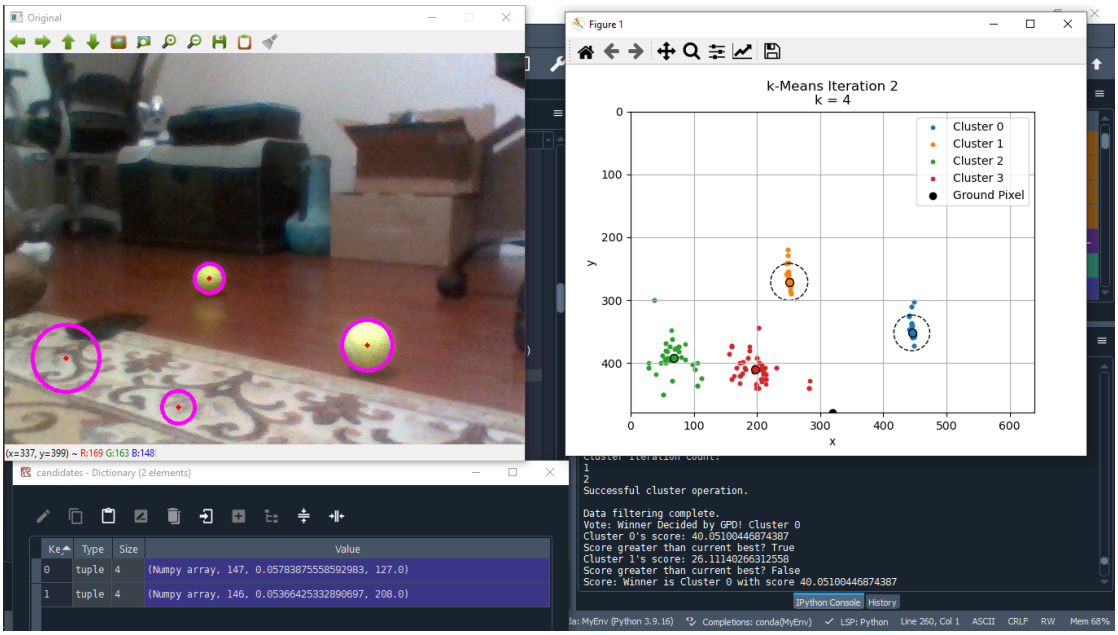


Figure 7: Voting Results - Cluster 0 wins.

3.1 Angle Calculation

Since the voting function returns the coordinates of the winning cluster's centroid, these coordinates are used to calculate the angle between the vector made from the ground pixel to the centroid $\vec{v}$ and the unit vector from the ground pixel in the vertical direction $\hat{u}$. So, a given cluster has an angle range of $[-\frac{\pi}{2}, \frac{\pi}{2}]$, where CCW (left half of the image) from the vertical is positive and CW from the vertical (right half of the image) is negative. Because the angle θ is measured from the vertical, it can be calculated by using the components of $\vec{v}$. Given centroid (i, j) and ground pixel a, b where a = image height and $b = \frac{\text{image width}}{2}$, the angle θ is found by:

$$\vec{v} = (i, j) - (a, b) = \begin{bmatrix} i - a \\ j - b \end{bmatrix} = \begin{bmatrix} v_y \\ v_x \end{bmatrix}$$

$$\theta = \boxed{\arctan\left(\frac{j-b}{i-a}\right) = \arctan\left(\frac{v_x}{v_y}\right)}$$

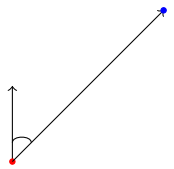


Figure 8: Angle Diagram; Centroid point is in blue. The ground pixel is red.

As a sanity check, I used OpenCV to rotate the image about the calculate angle during a test and received the results in Figure 9. The detected tennis ball (center) is aligned with the image center vertical, which is the desired test result.
With the completion of the parameter tuning, cluster selection, and angle calculation in the test environment, I then moved to implement the program in ROS.

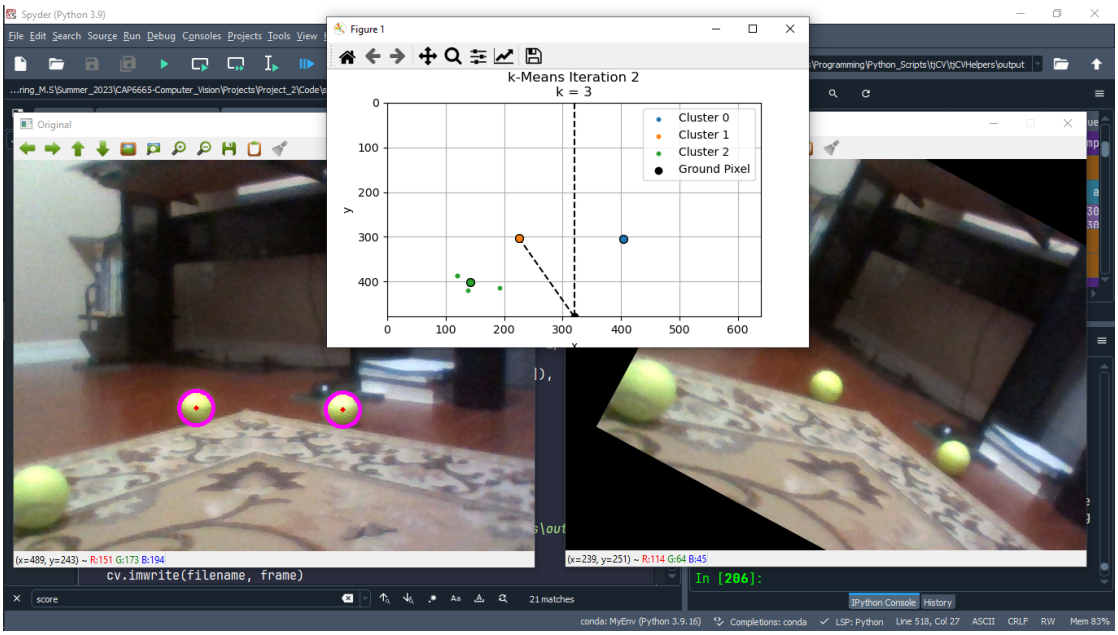


Figure 9: Angle Correction - Middle ball becomes aligned with the center vertical.

4 ROS Implementation

4.1 Structure

4.2 Tests

4.3 Package Construction

4.3.1 numpy_msgs Package

4.3.2 Main Package

5 Flowchart

6 Conclusion

A Acceleration Plots

This section presents more acceleration plots that would have cluttered the report but are very helpful for understanding.

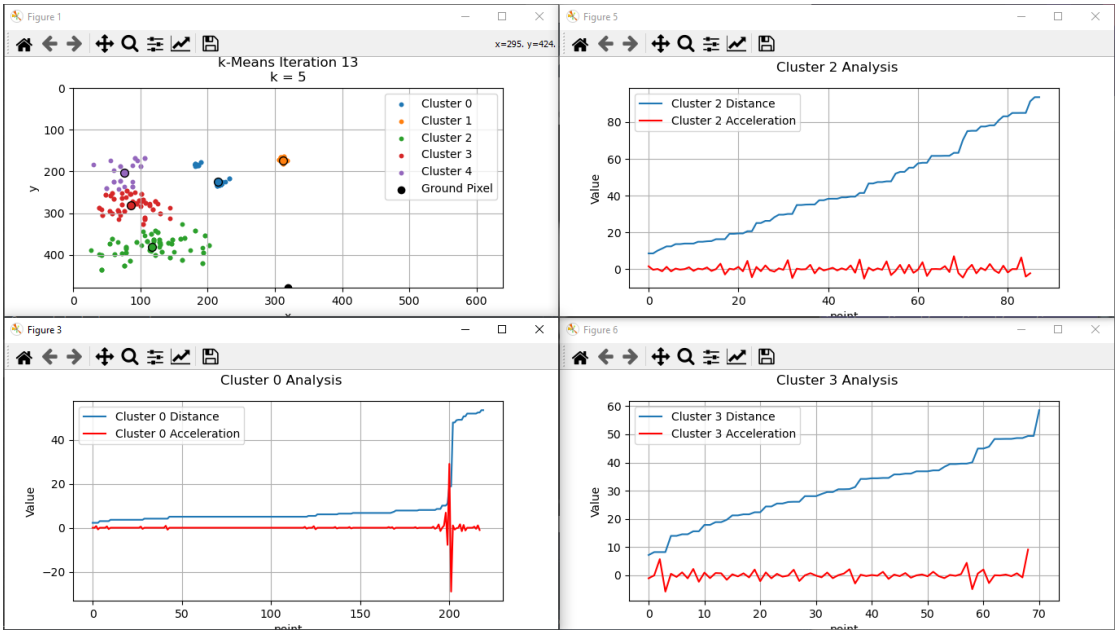


Figure A.1: Showing the plots for three clusters. Useful to show that clusters with a lot of spread (noise clusters) also have low acceleration, adding validity to the method for seeking legitimate clusters.

Figure A.2 shows the acceleration plots for three clusters. The 'good' cluster, Cluster 0, has a max acceleration of approximately two. Cluster 2, a bad cluster due to it being noise, also has high acceleration, but it missed the minimum point count, so it was not a candidate for consideration. Even if it did enter candidacy, it would lose because even after adjusting for outlier points, its minimum centroidal distance is 20, which would result in low density. Cluster 1, however, is a candidate for consideration and it has high acceleration, meaning there is at least one outlier point.

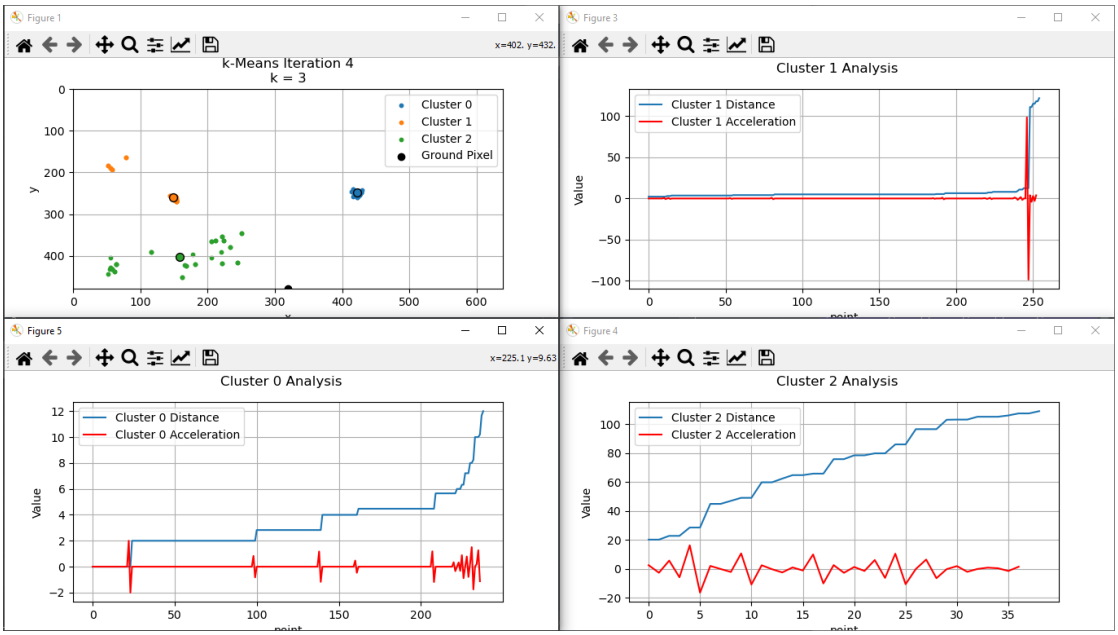
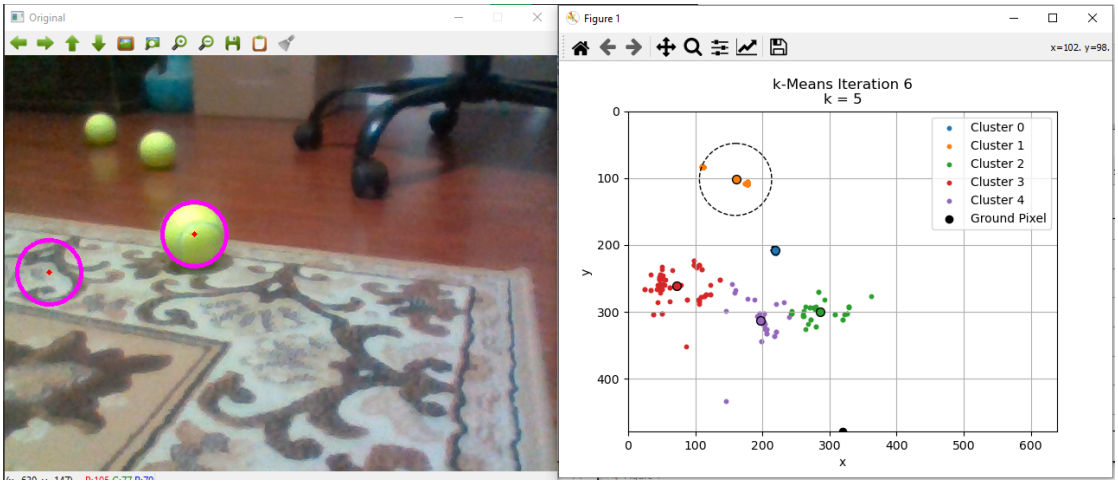


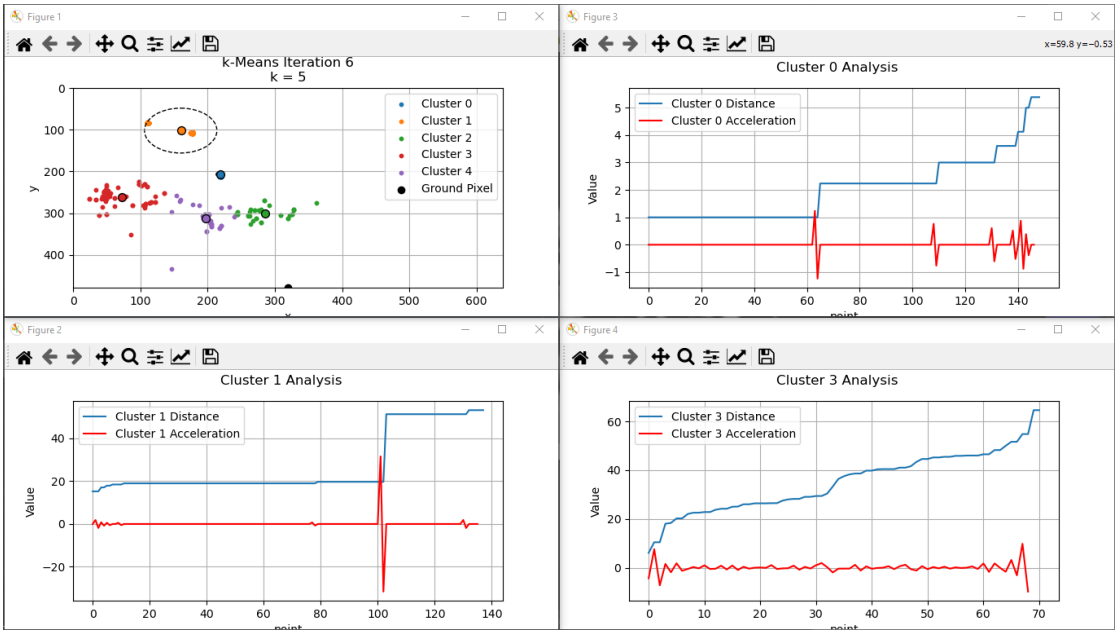
Figure A.2: Showing the plots for three clusters. Cluster 1 has high acceleration while Clusters 0 does not.

Figure A.3 shows an image along with the acceleration plots for selected clusters. Cluster 1 is the middle tennis ball. The outlier points of this cluster are the result of the left-most ball's occasional detection over the detection period. As it did not appear on the max detections frame, it was included in the middle ball's cluster. This is not an issue since Cluster 0 is the most valid ball to choose and the left-most ball is far away in terms of GPD.

One can also see that as a result of the outliers, Cluster 1 has a high acceleration, but Cluster 3, which is much more spread out due to being a noise cluster, does not.



(a) Matching Image



(b) Acceleration Plots

Figure A.3: Another Acceleration Plot. Notice the spread in Cluster 3 and its low acceleration compared to Cluster 1.