Project -3
2-D Object Detection
Arun Madhusudhanan

Project Description

The objective of the project is to detect 2D objects placed on a white surface in a translation, scale, and rotation invariant manner. The system created for this purpose has two modes: training and classification. In the training mode, the system will learn from a collection of images of objects with white backgrounds taken by a camera positioned directly above. The system will capture translational, scale, and rotational invariant features from these trained images and store them in a csv file. In the classification mode, the system will use nearest neighbor or k nearest neighbor methods, based on user input, to compare the characteristics of the target object with the features in the database to find the closest match. If the closest match is not found, the system will switch back to training mode to add the unknown object to the database, based on the user's input.

Setup

I couldn't obtain a webcam, and when I attempted to stream through my phone, there was a significant amount of lag. Hence the system is developed to read images from a directory for training and classification. The photos of objects were taken using a smartphone positioned directly above.

Tasks Performed

1. Thresholding image

The images were preprocessed before thresholding. First, a 3*3 Gaussian filter was used to blur the image. Next, in the HSV space, the intensity value of colors with a saturation greater than a value of 70 was reduced by 30%. If the sum of the RGB pixel values in the processed image was greater than 300, the pixel values of the single channel output were labeled as background and set to 0 (white color). If the sum of the RGB pixel values was less than 300, the pixel values were regarded as belonging to the foreground.

Examples for 3 thresholded objects are provided in figure 1 to 3.

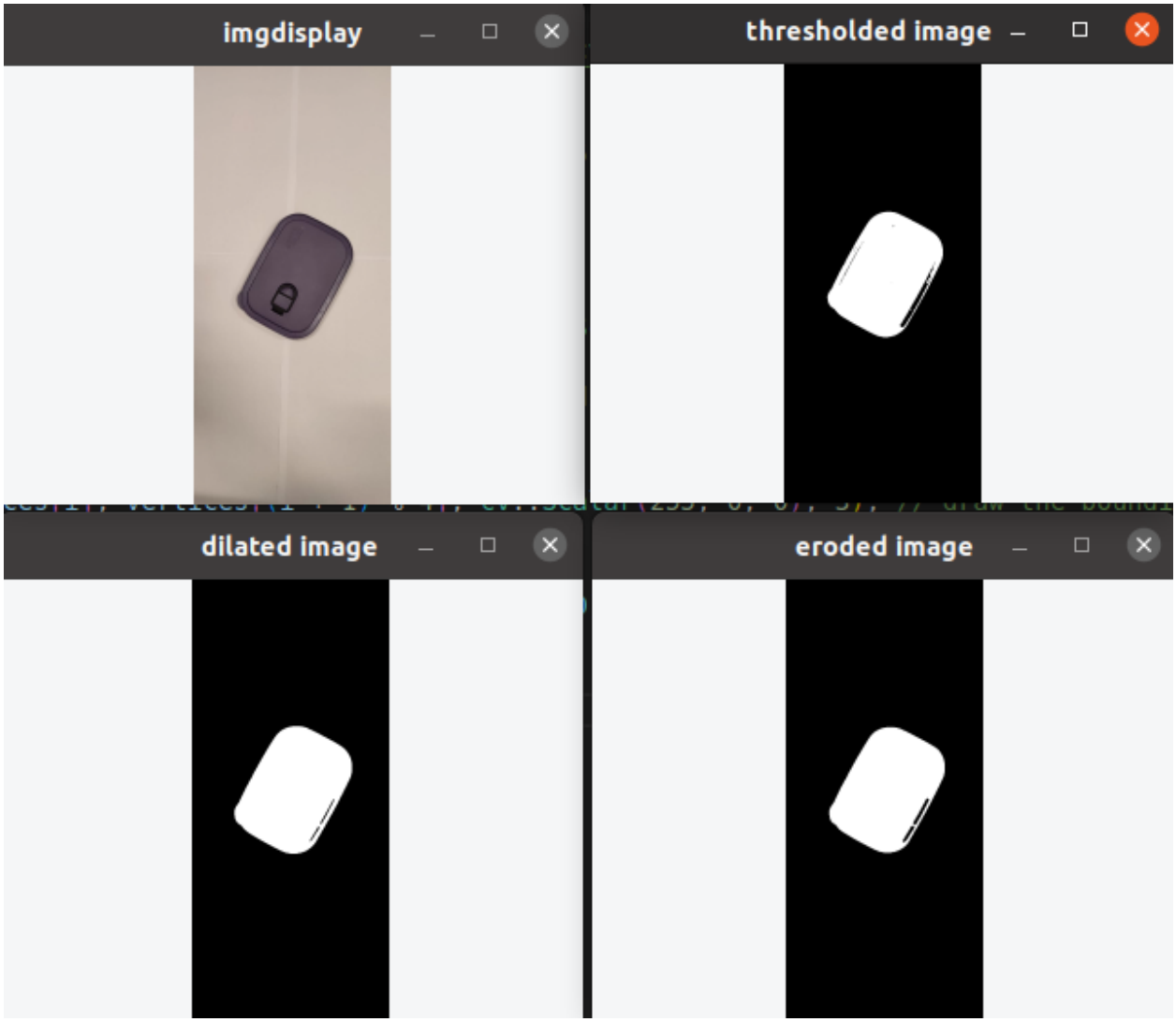


Figure 1: Object- Lid (order of images : raw image, thresholded image, dilated image and eroded image)

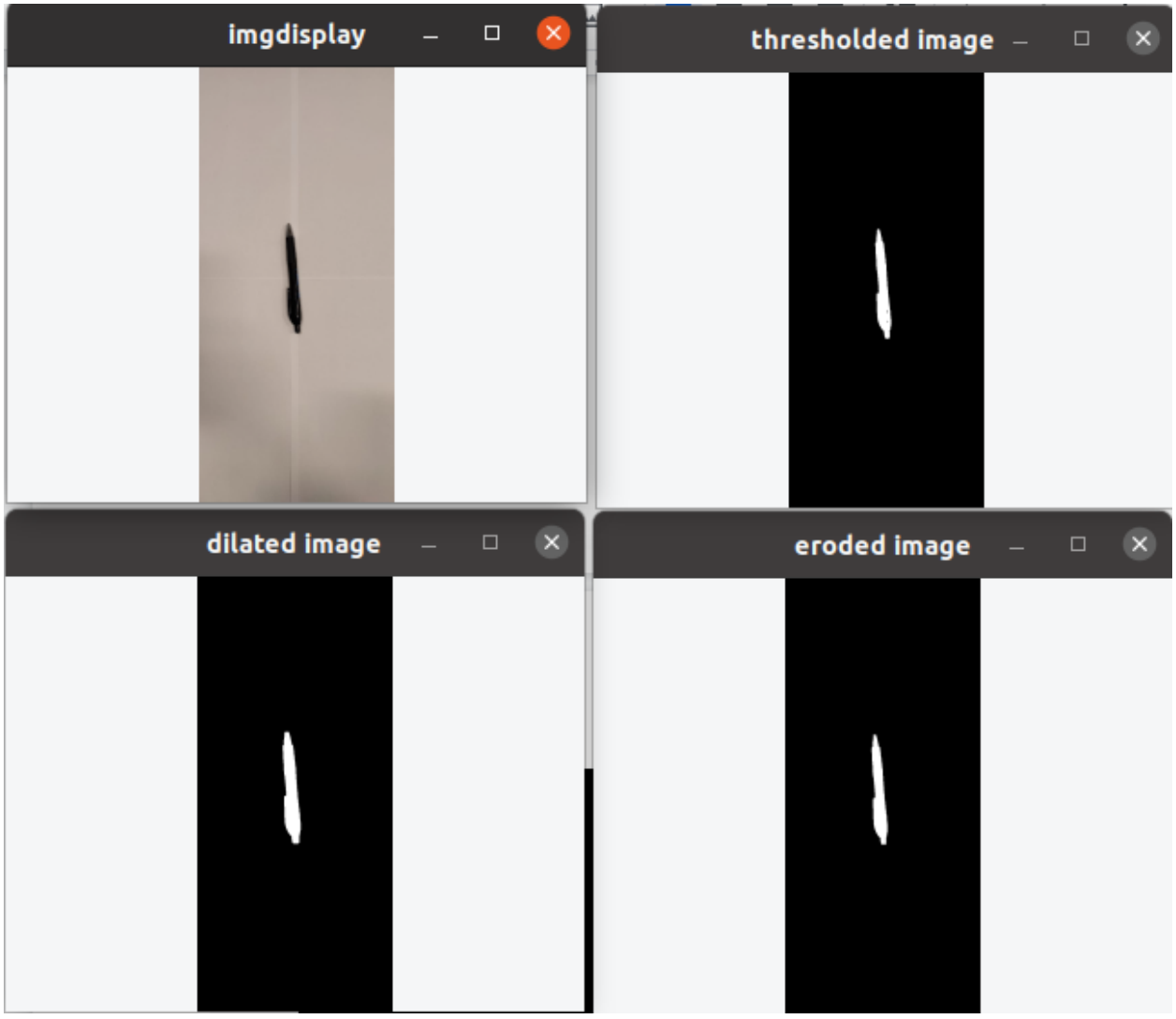


Figure 2: Object - Pen (order of images : raw image, thresholded image, dilated image and eroded image)

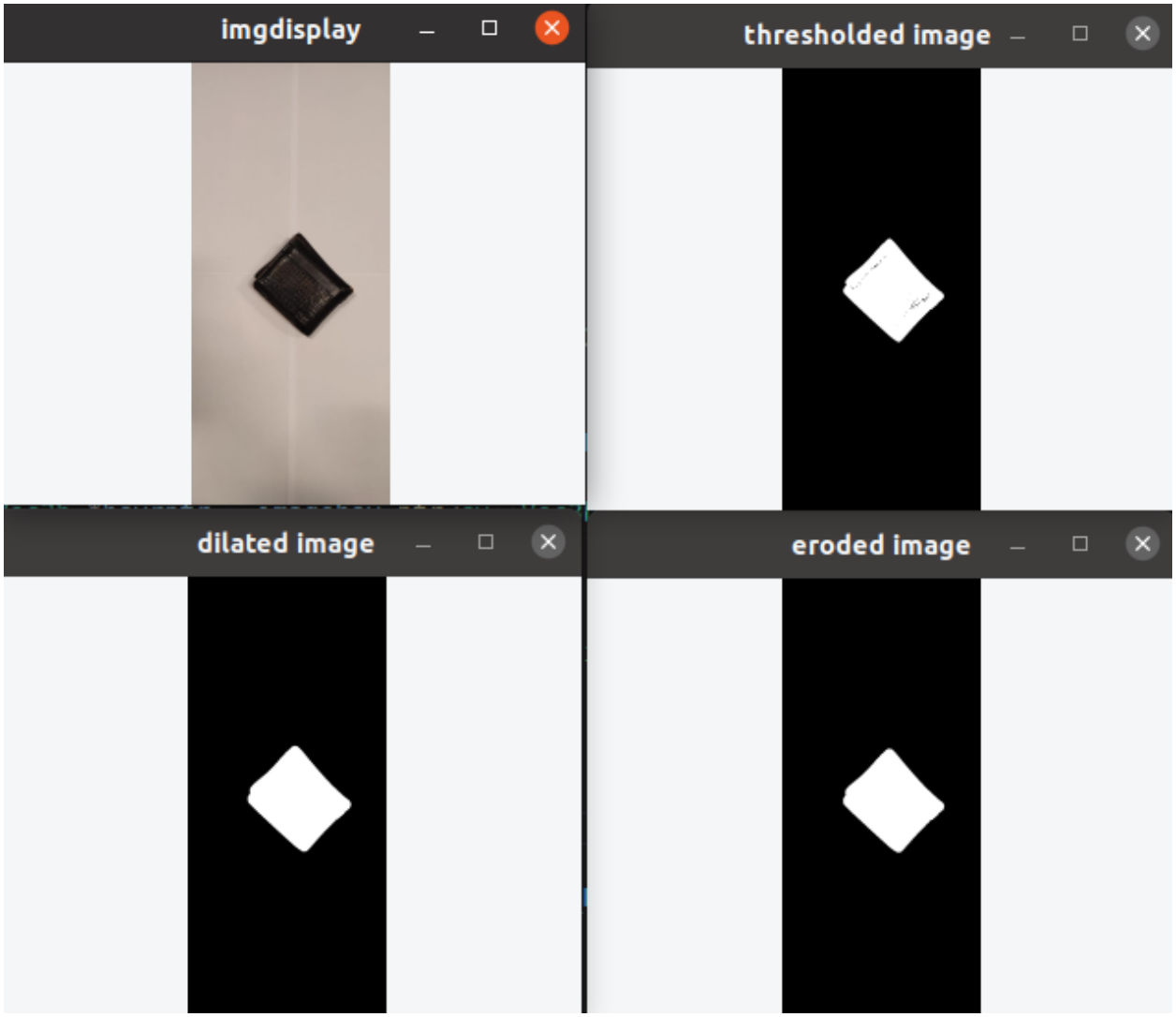


Figure 3: Object - Wallet (order of images : raw image, thresholded image, dilated image and eroded image)

2.Clean up the binary image

To refine the thresholded images, a morphological filter was applied. The images were first dilated with 8 connected neighbors for 3 iterations. Following that, they were eroded with 4 connected neighbors for 3 iterations. The reason for performing the dilation with 8 connectedness first was that the thresholded images in the training dataset had mostly holes. Dilation fills the holes. To preserve the original image shape, the erosion with 4 connectedness was done for 3 iterations, which is considered best practice. Figure 1-3 shown above displays the outcomes of the morphological filtering process.

3.Segment the image into regions

I used inbuilt connected components with stats function in OpenCv for the connected component analysis. The areas with more than a threshold pixel count of 2000 are considered as valid regions. Then the valid regions are sorted based on the decreasing order of pixels. Only top 3 regions will be detected as shown in figure 7.

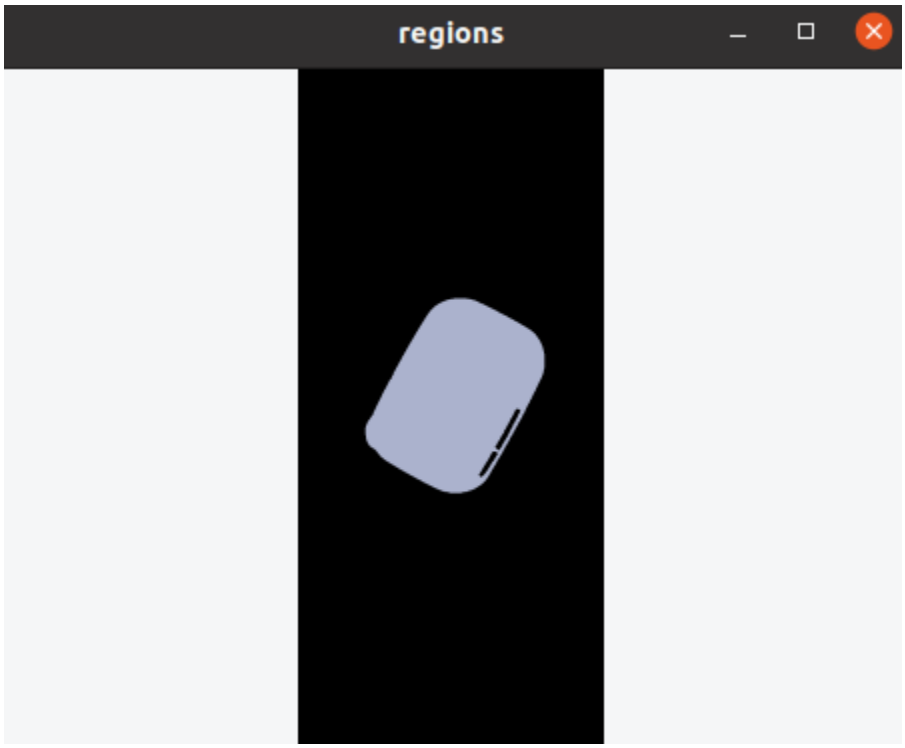


Figure 4: Region map of Lid

Figure 5: Region map of Pen

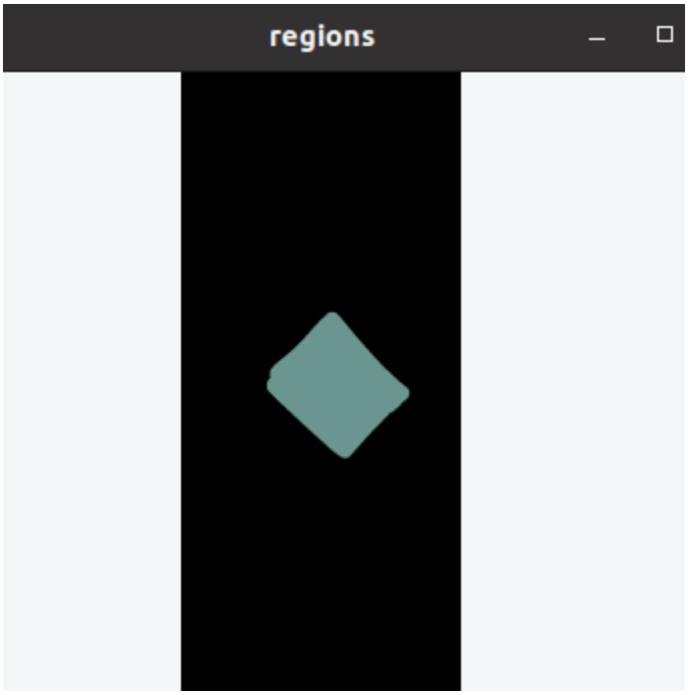
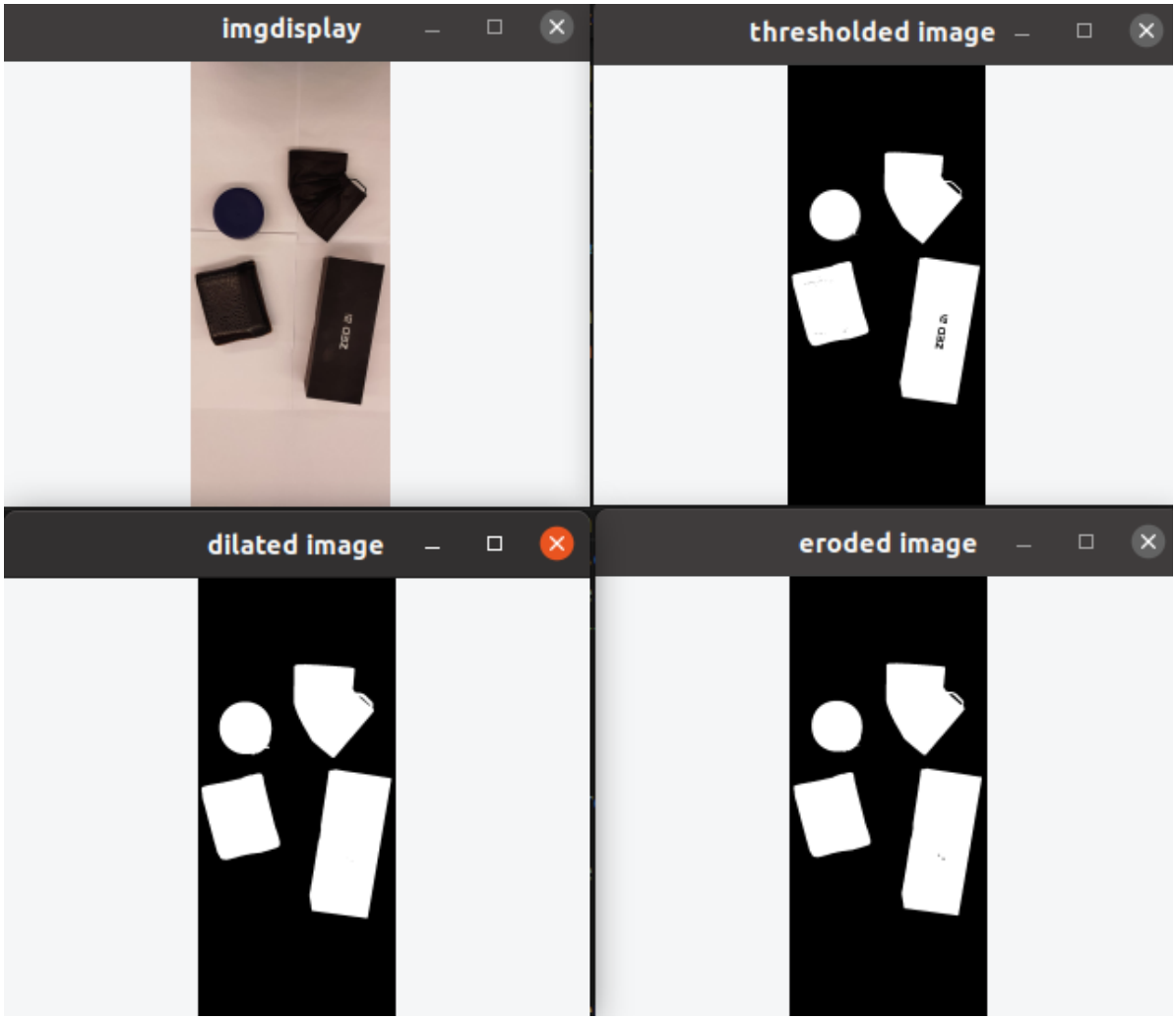


Figure 6: Region map of Wallet


imgdisplay
thresholded image
dilated image
eroded image


regions

Figure 7: Multiple objects (wallet, mask, ZED box, bottle cap) at a time. As explained the system is able to detect multiple regions at a time but only the top 3 regions are considered as shown in the figure 7. Bottle cap with the smallest area is neglected.

4. Compute features for each major region

I used the OpenCv moments() function to calculate moments for finding the axis of least central box and bounding box of the objects.

Moments() function gives raw moments, central moments (which are translational invariant),normalized central moments (which are translational and scale invariant). [2]

Raw moments are calculated using the following equation 1, where i and j are integers, *I* is the pixel intensity at location (x,y)

$$M_{ij} = \sum_{x} \sum_{y} x^i y^j I(x, y) \tag{1}$$

The centroid of the regions can be found using the equation (2)

$$\bar{x} = \frac{M_{10}}{M_{00}}$$
$$\bar{y} = \frac{M_{01}}{M_{00}} \tag{2}$$

Central moments which are translational invariants can be found using the equation 3.

$$\mu_{ij} = \sum_{x} \sum_{y} (x - \bar{x})^i (y - \bar{y})^j I(x, y) \tag{3}$$

The normalized central moments which are scale invariant can be found using the following equation

$$\eta_{ij} = \frac{\mu_{i,j}}{\mu_{00}^{(i+j)/2+1}} \tag{4}$$

However these moments are not rotational invariant. Hence Hu Moment are considered.

Hu Moments are a collection of seven numerical values that are calculated using central moments and are unaffected by changes made to an image. The initial six moments have been demonstrated to be unaffected by transformations such as translation, scaling, rotation, and reflection. However, the seventh moment changes its sign when an image is reflected[2]. Hu Moments are calculated using the equation 5.

$$
\begin{aligned}
h_0 &= \eta_{20} + \eta_{02} \\
h_1 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
h_2 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
h_3 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
h_4 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
h_5 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})] \\
h_6 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned}
\quad (5)
$$

I used the in-built HuMoments function in OpenCV. Following are the rotational, translational, scale invariant features I used .

- First four Hu Moments
- Height to width ratio of bounding box
- Percentage of bounding box filled.

The system will show one feature at the centroid during this step. I selected it as Height to width ratio of the bounding box.

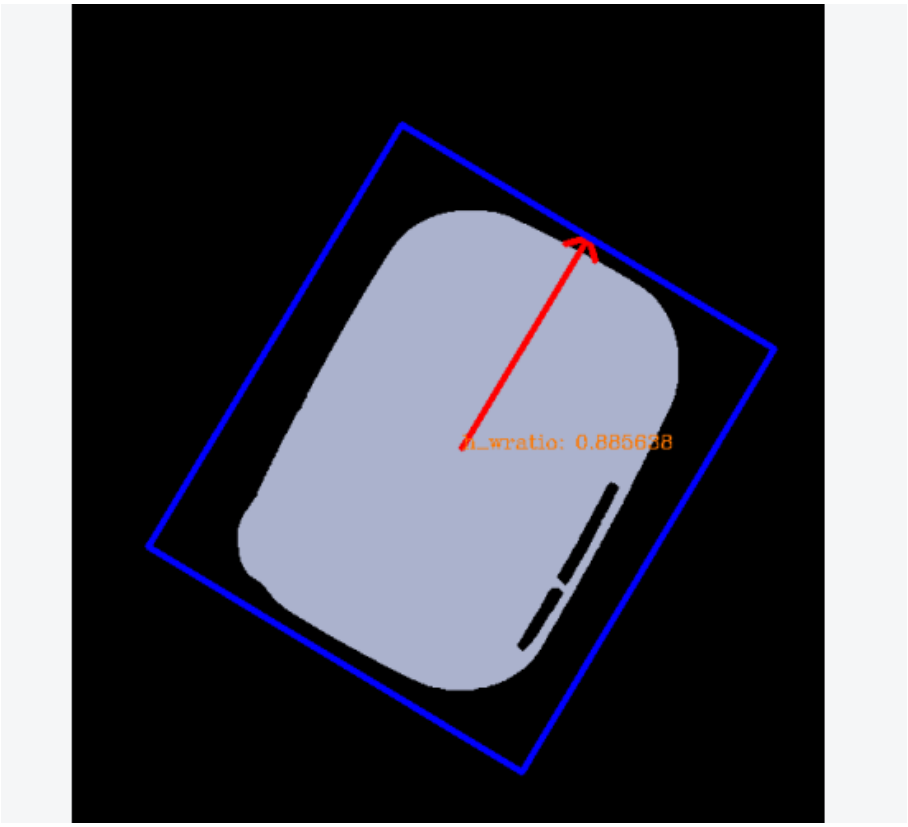


Figure 8: Region map of Lid with the axis of least central moment and the oriented bounding box

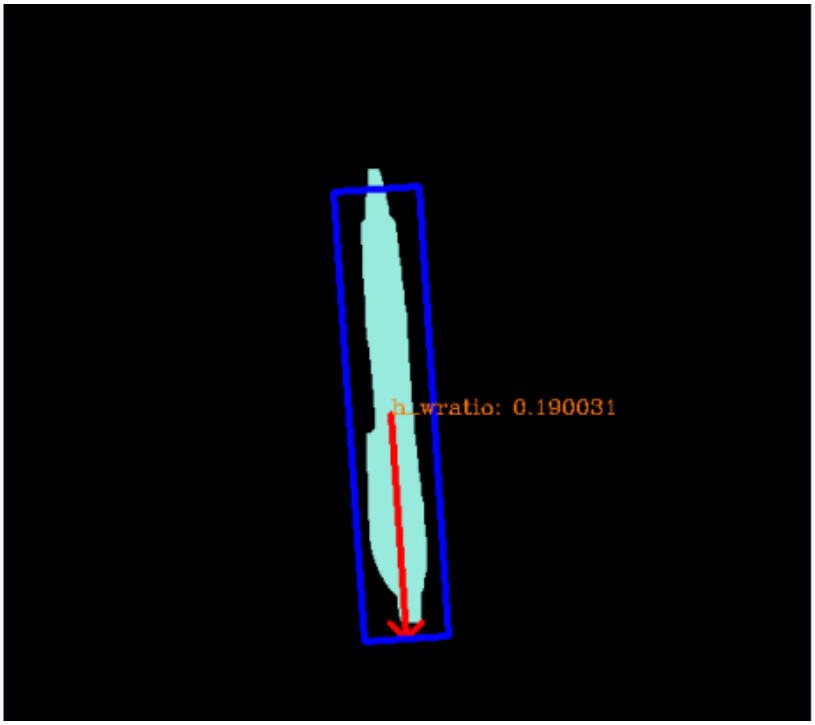


Figure 9: Region map of Pen with the axis of least central moment and the oriented bounding box

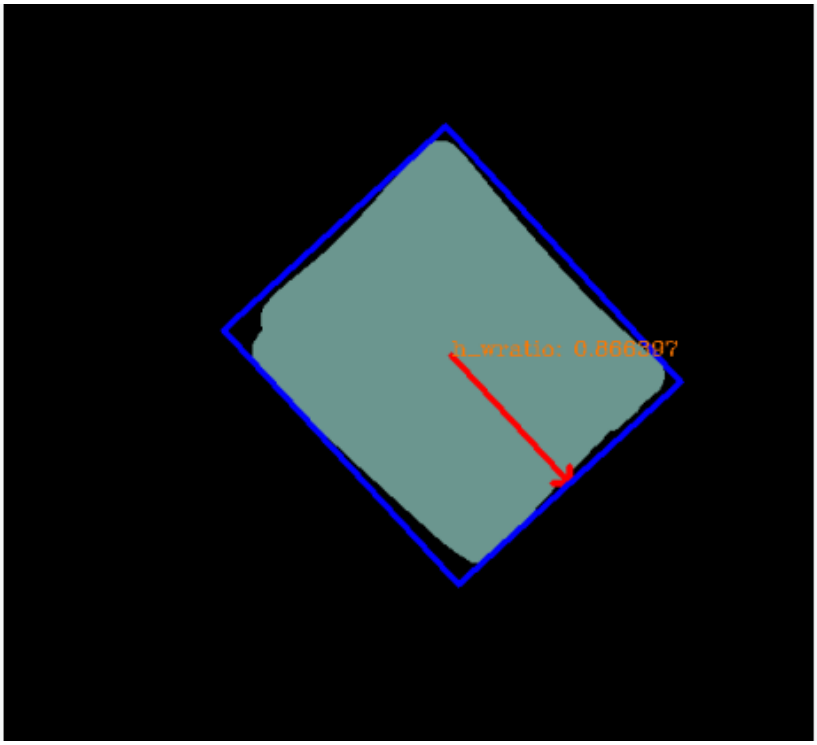


Figure 10: Region map of Wallet with the axis of least central moment and the oriented bounding box.

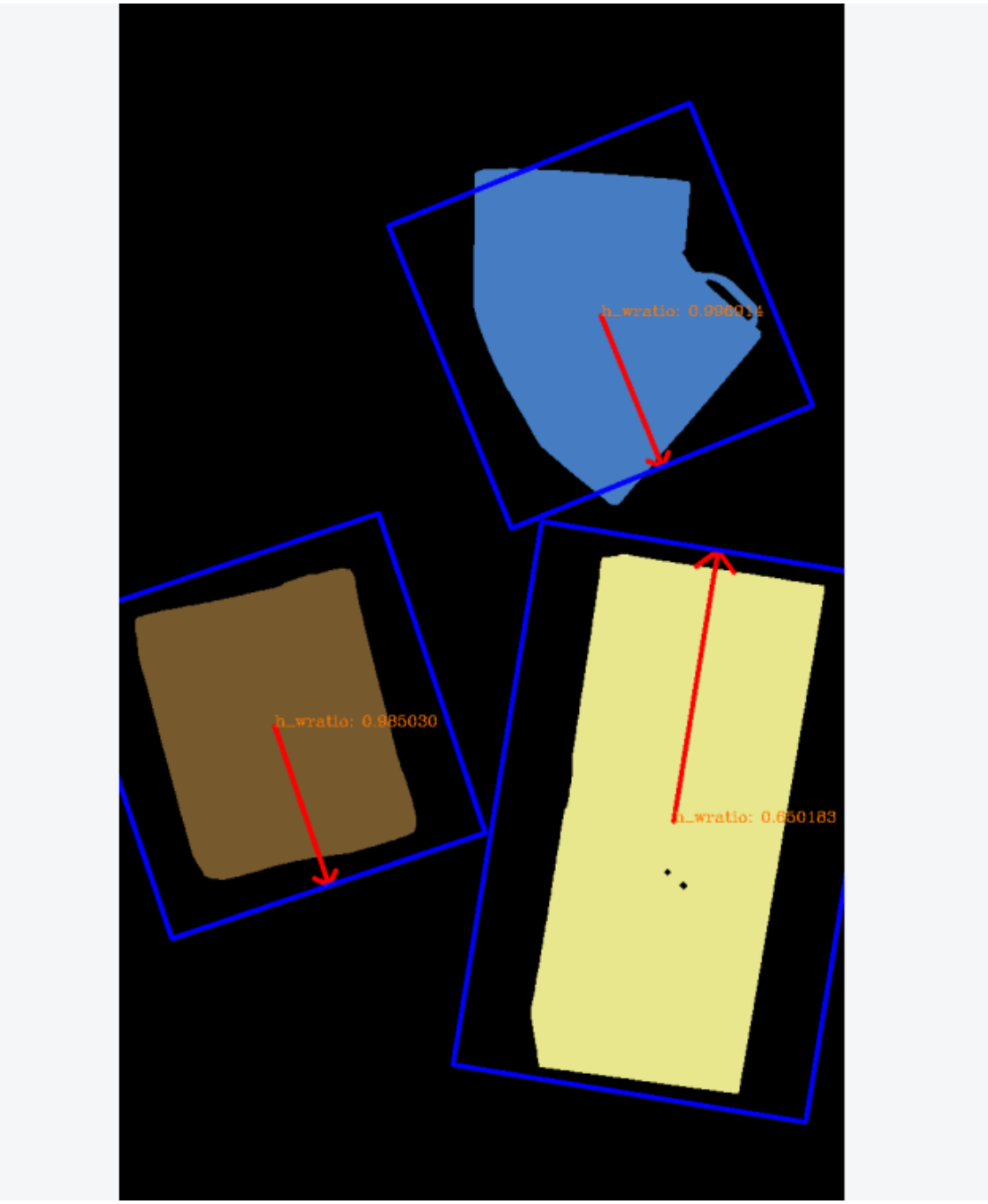


Figure 11: Region map of multiple objects (wallet, mask, ZED box, bottle cap) at a time with the axis of least central moment and the oriented bounding box. As explained the system is able to detect multiple regions at a time but only the top 3 regions are considered. Out of 4, only the top 3 regions are considered. Bottle cap with the smallest area is neglected.

5. Collect training data

In the training mode, the system will learn from a collection of images of objects with white backgrounds taken by a camera positioned directly above. The images were taken in a smartphone and stored in a directory as explained in the set up section. The system will capture translational, scale, and rotational invariant characteristics from these trained images and store them in a csv file.

Users can opt the training mode by pressing ‘t’ when the system prompts for input.
Thirteen objects were considered for training.

- Lid
- Airpod case
- Beer opener
- Pen
- Wallet
- Watch
- Cap
- Spoon
- Glass case
- Zed
- Packet clip
- Cloth
- Mask

The labels for a few objects were already saved in the system as shown in the figure 12. Users will just have to enter the label while training.

```
std::map<char, std::string> label_map {
            {'p', "pen"}, {'l', "lid"}, {'z', "zed"}, {'g', "glasscase"},
             {'c', "calculator"}, {'b', "beeropener"},
            {'m', "mask"}, {'i', "packet clip"},{'o', "comb"},{'r', "creditcard"},{'k', "key"},
            {'w', "wallet"},{'a', "airpodcase"}
};
```

Figure 12: Label maps for different object classes

If at all, the user is entering a new label, the system will prompt the user to enter the new label name and class name via the command prompt. This functionality is later used while training an unknown object.

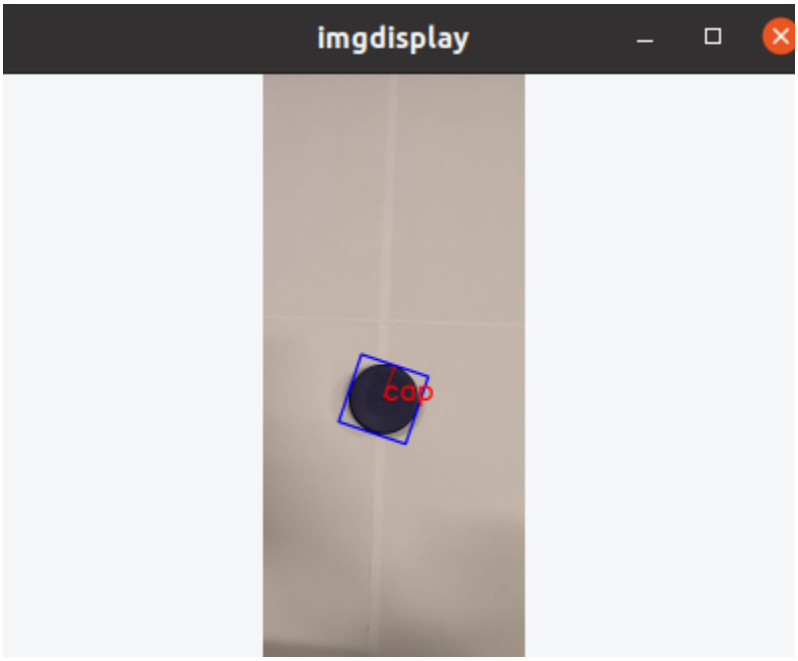
Six different images of an object were captured using a smartphone, out of which three were used for training and remaining for inference.

6. Classify new images

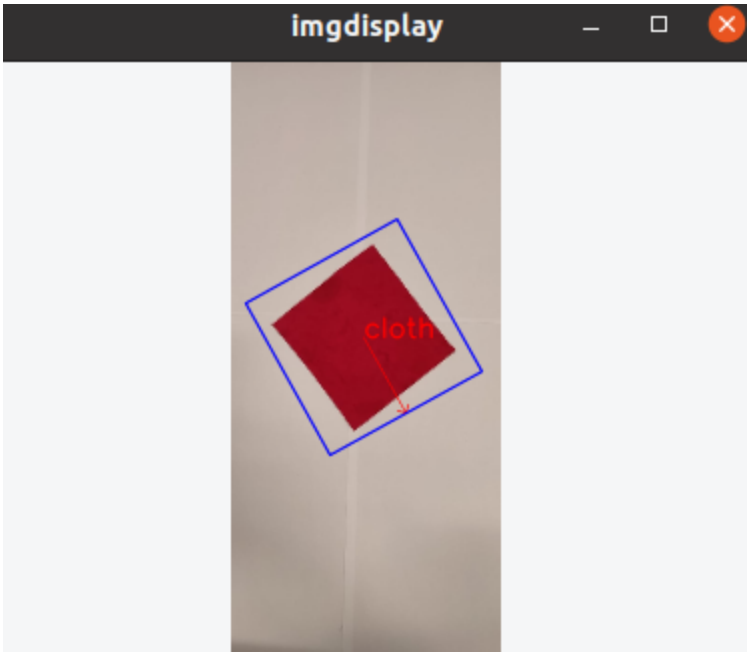
Three different images of each object were used for classification. An object is identified correctly at least two times as shown in the confusion matrix later.

I used a scaled Euclidean distance metric [sum of {sqrt ((x_1 - x_2)^2)} / stdev_x] to classify new images. The standard deviations were calculated for each feature across all classes in the database.

The database object with the smallest distance is considered as the nearest neighbor. The results from at least one image of each object with the assigned label is shown in figure 13.

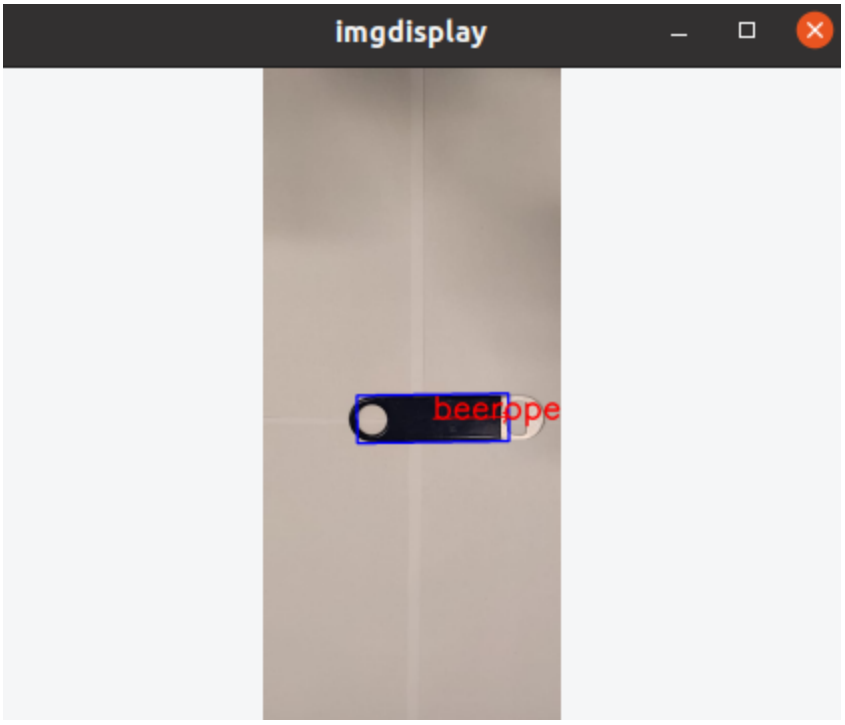


Object 1: Bottle Cap

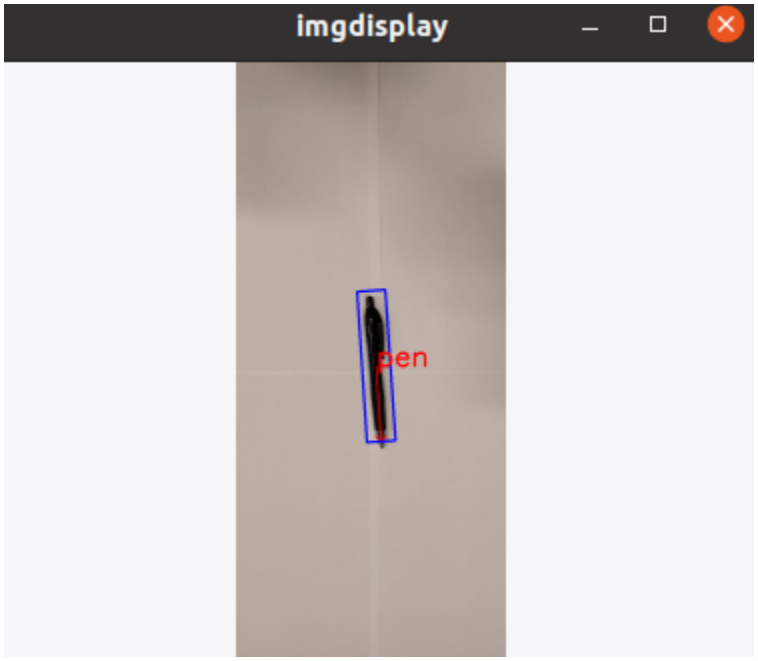


Object 2 : Cloth

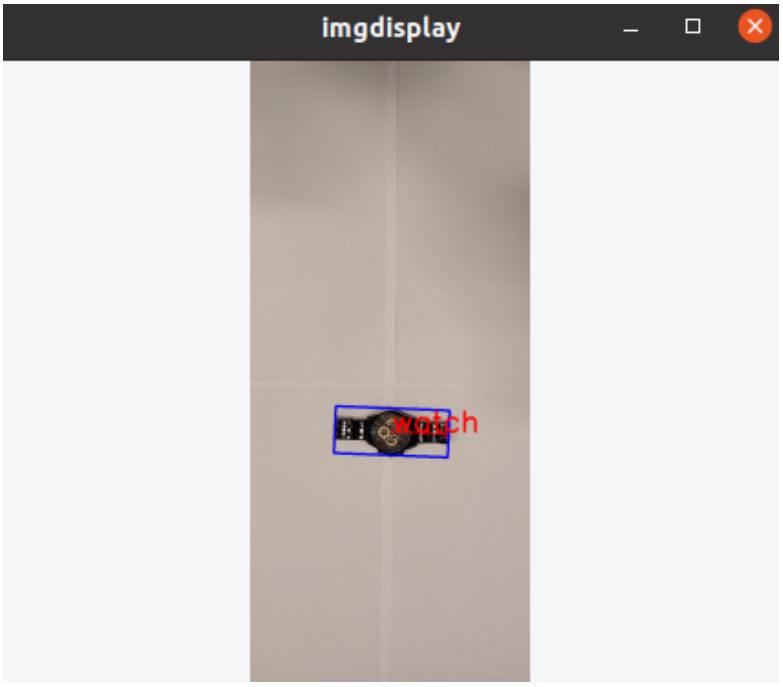
Object 3: Mask



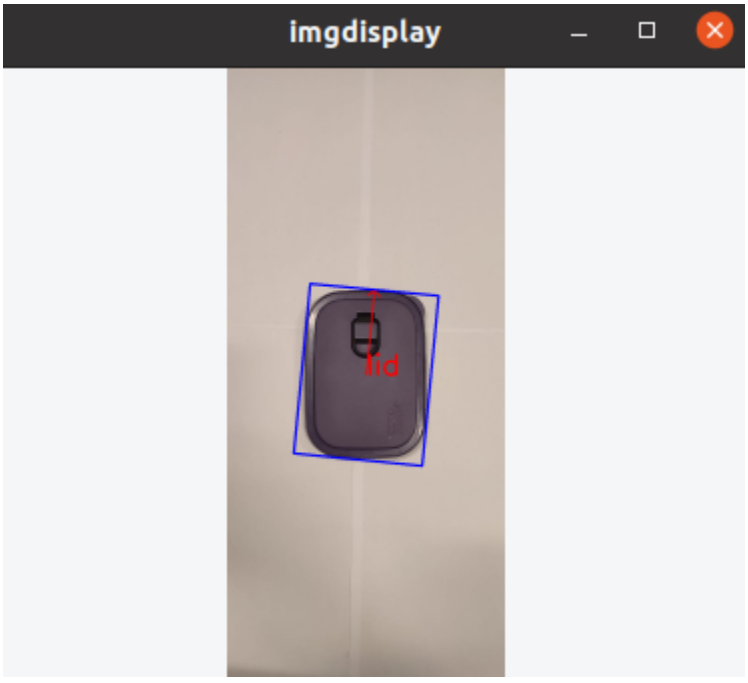
Object 4: Beer Opener



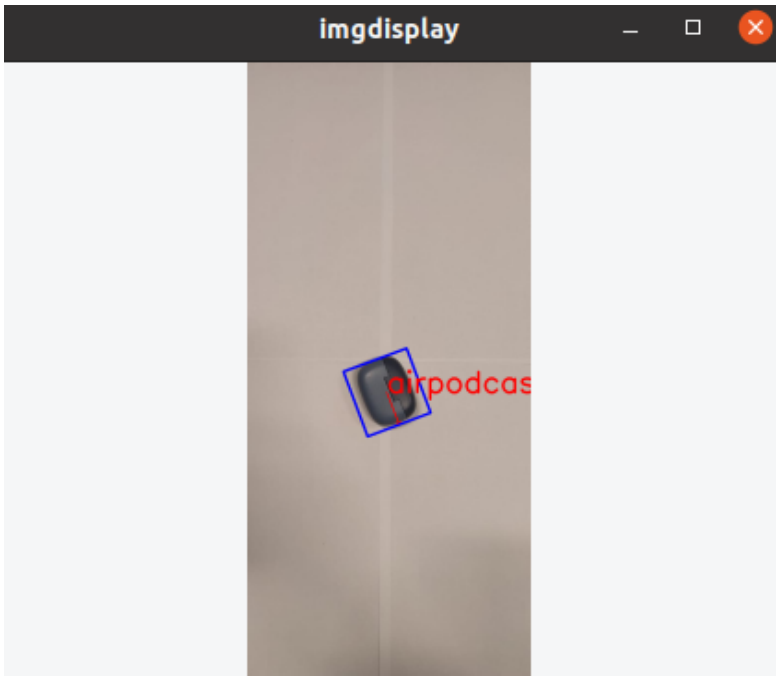
Object 5: Pen



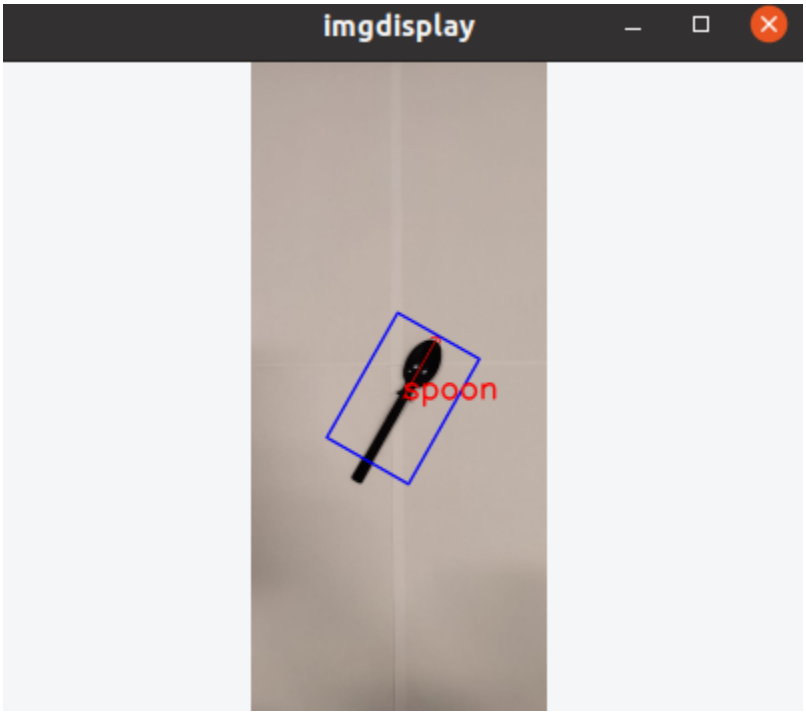
Object 6: Watch



Object 7: Lid

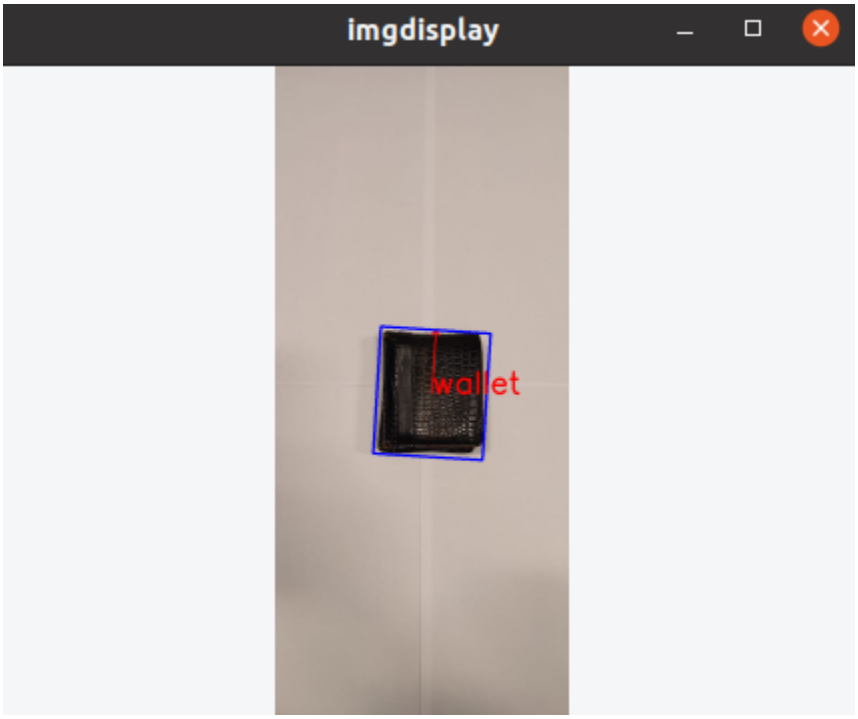


Object 8: Airpod Case

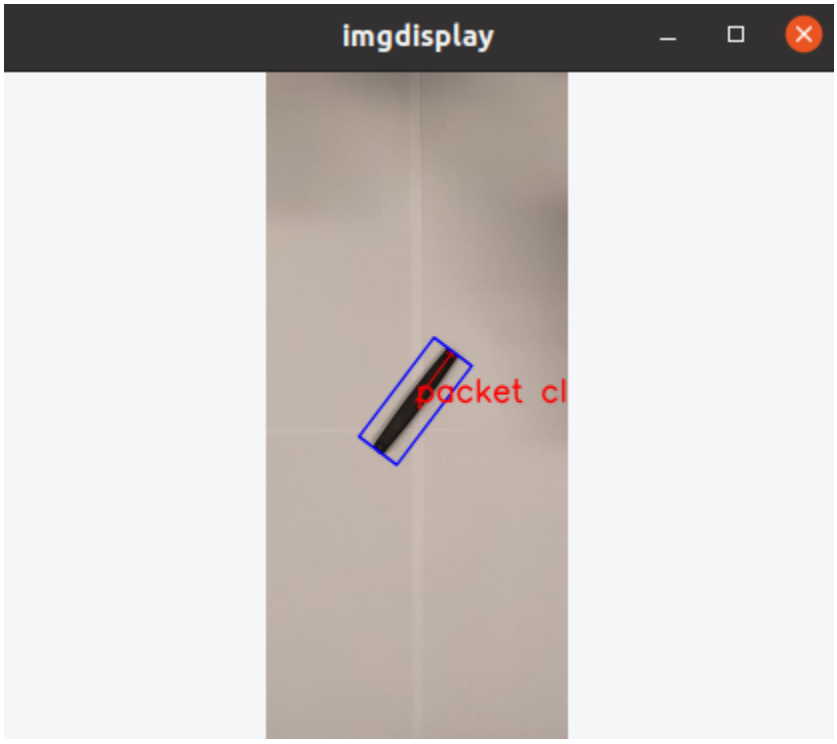


Object 9: Spoon

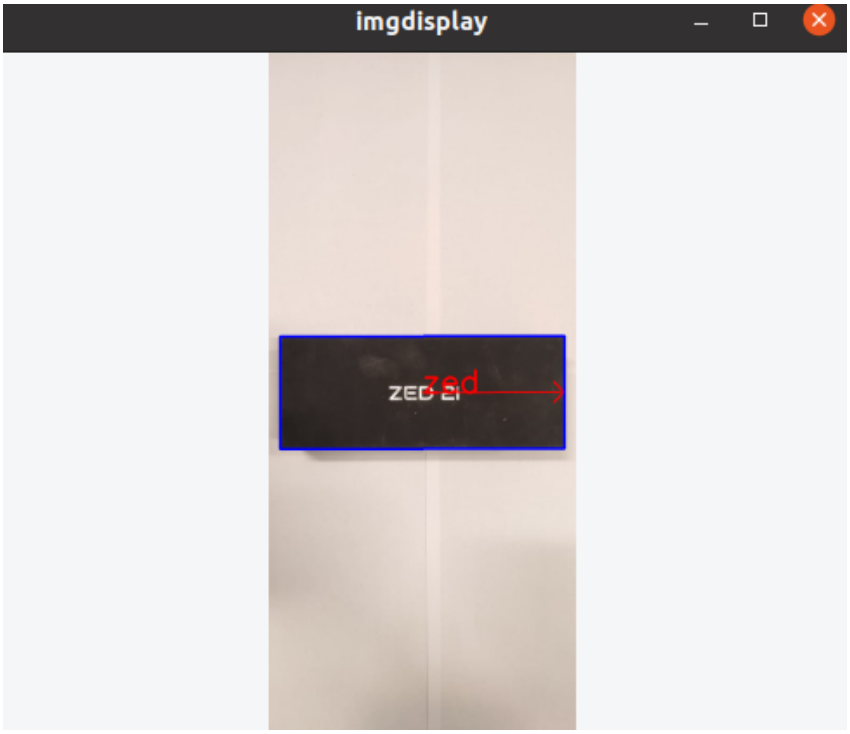
Object 10: Glasses(spectacles) case



Object 11: Wallet



Object 12: Packet clip



Object 13: Zed Camera box

Figure 13: Classified images

7. Implement a different classifier

I used K - Nearest Neighbour matching with K = 3 as another classifier. Following steps are done for KNN approach

- A minimum of 3 images for the same classes were trained.
- Find the scaled Euclidean distance between target feature and database features.
- Mean of the 3 shortest Euclidean distance to image features of each class were calculated. This is considered as the distance between the target object and classes in the database
- The class with the smallest mean was identified as the similar object.

An object is identified correctly at least two times as shown in the confusion matrix later. The figures for similar matches with bounding boxes with labels were similar to that shown under section ‘Classify new images’.

8. Confusion Matrix

The performance of the system was evaluated using a confusion matrix(shown in figure 14 and 15).

		OUTPUT											
		lid	airpodcase	beeropener	pen	wallet	watch	cap	spoon	glasscase	zed	packet clip	cloth	mask
INPUT	lid	2	1											
	airpodcase	1	2											
	beeropener			3										
	pen				3									
	wallet					2								1
	watch						3							
	cap							3						
	spoon				1				2					
	glasscase									3				
	zed										3			
	packet clip											3		
	cloth												3	
	mask													3

Figure 14: Confusion matrix for nearest neighbor match

The nearest neighbor cannot find the lid, airpod case, wallet,spoon accurately at least once. The wallet was classified as a mask once and a spoon was classified as a pen once. This might be due to the fact that they have similar shapes and sizes. However, detecting the airpod case for lid and viceversa seems odd. I think this might be due to some factors/errors happened while capturing data.

		OUTPUT											
		lid	airpodcase	beeropener	pen	wallet	watch	cap	spoon	glasscase	zed	packet clip	cloth	mask
INPUT	lid	3												
	airpodcase	1	2											
	beeropener			2						1				
	pen				2				1					
	wallet					2								1
	watch					1	2							
	cap							2					1	
	spoon				1				2					
	glasscase									3				
	zed										3			
	packet clip											3		
	cloth												3	
	mask													3

Figure 15: Confusion matrix for K nearest neighbor match

The K nearest neighbor cannot find the airpod case, beer opener, pen, watch, wallet, cap(bottle), spoon accurately at least once. This might be due to the fact that some objects considered have similar shapes and sizes.

As shown the objects were detected correctly at least 2 times for all classes.
The accuracy for nearest neighbor matches is (35/39) = 89.7%.
The accuracy for K nearest neighbor matches is (32/39) = 82.05%.

The accuracy for nearest neighbor approach is found to be higher than the K nearest neighbour in this case. This might be due to the nearest neighbor only considering the minimum distance for matching classes, where KNN considers the mean distance for each class. So if one image of an object class has some anomaly, it will affect the final result.

9 : Demo

Training (sample video)
https://drive.google.com/file/d/1spLJT8_ohKMc-Lj54QVOOqUwuiArh2EE/view?usp=share_link

Nearest Neighbour classification (full video):
https://drive.google.com/file/d/1NGfmw2nyGvinInC6ZS4FDD1Y4LxdSwwq/view?usp=sharing

K-NN classification (sample) :
https://drive.google.com/file/d/1JlzFAC6rj2VSe3HrLpcWrVTj5UpQaBO9/view?usp=sharing

10 : Extensions

1. Can classify more than 10 objects as shown in task 6.
2. Can detect multiple objects at a time. For example, both masks and cap were identified correctly when they were captured in one single picture.

Figure 16: Multiple object detection

3. Implementing 2 pass connected component analysis from scratch. This is in addition to developing the thresholding method from scratch. I coded union - find approach from scratch. The results of 2 pass connected component analysis for region detection was compared with that of inbuilt function in OpenCV as shown in figures 17 and 18. The results of the custom function look very satisfactory.

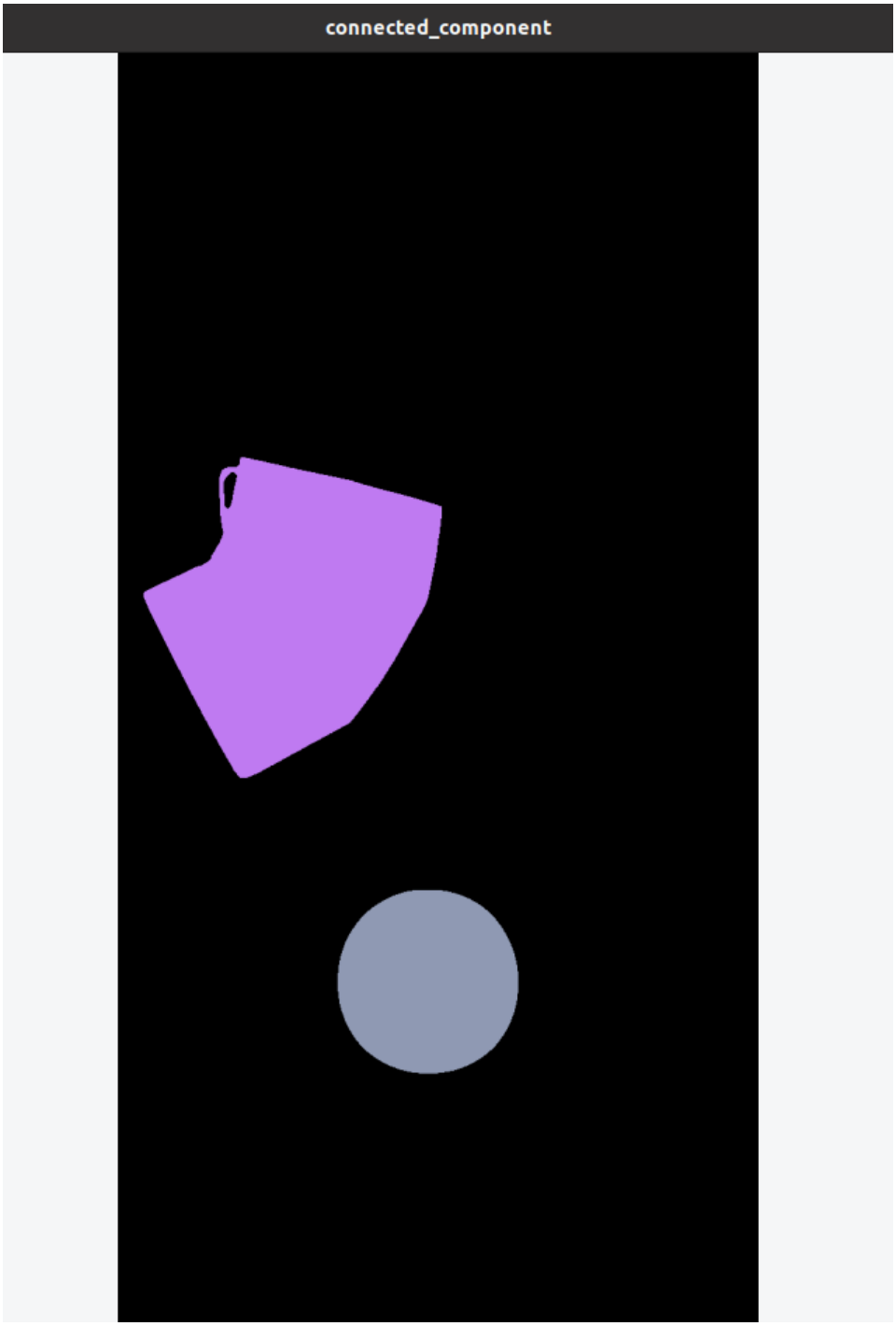


Figure 17: Results of 2 pass connected component method

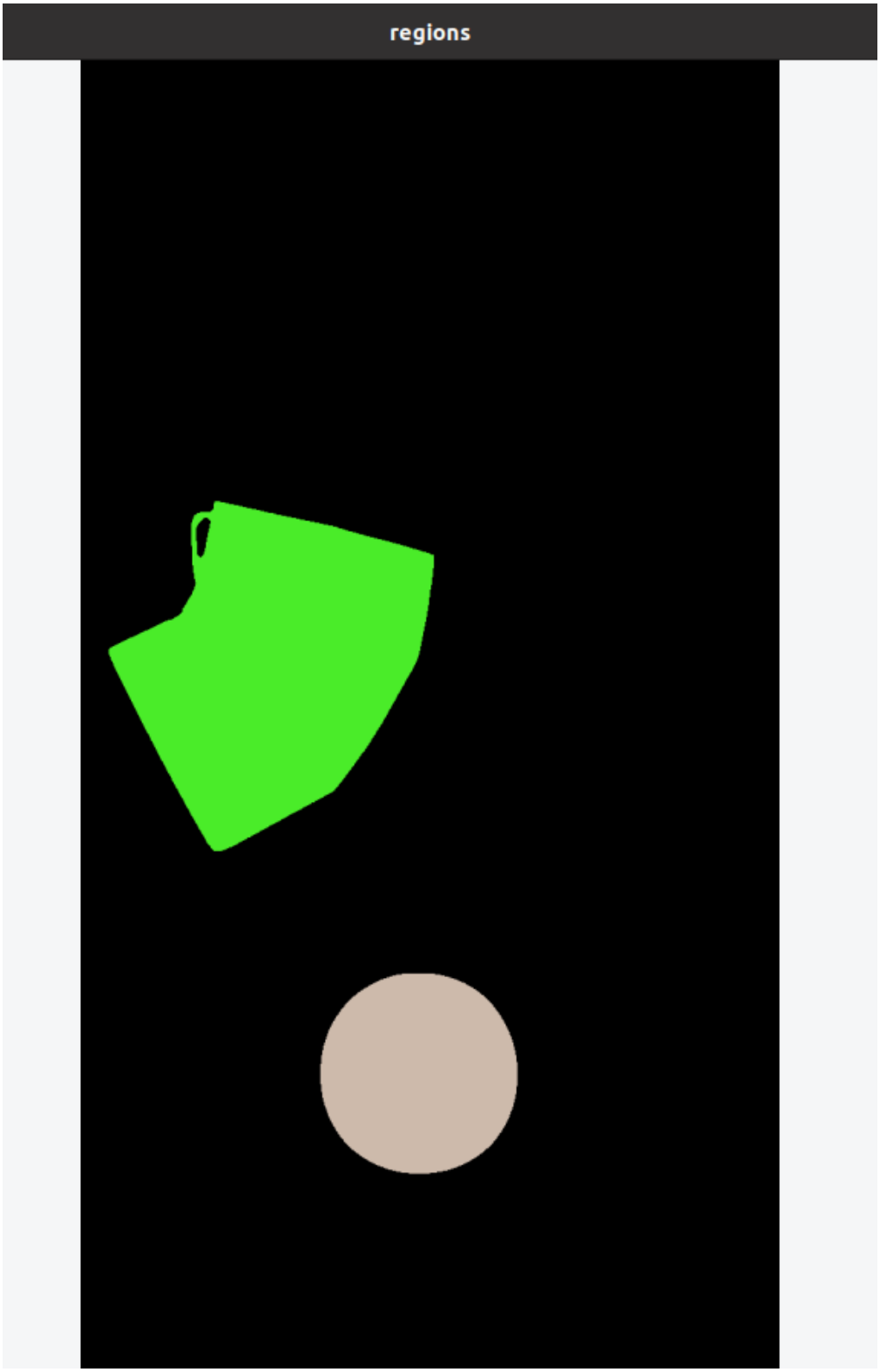


Figure 18: Results of inbuilt connected components with stats function in OpenCv

4. System identifies an object which is not in the database as unknown and goes into training mode. Demo: https://drive.google.com/file/d/1NGfmw2nyGvinlnC6ZS4FDD1Y4LxdSwwq/view?usp=sharing

Lessons Learned:

I learned about thresholding, morphological filters, connected component analysis (segmentation) tasks required for 2D object detection. I learned how important data collection is for a computer vision problem. I learned about nearest neighbor approaches, nearest approaches and their differences. I got to learn about union, find data structure and map data structure in cpp. I also got familiarized with some in-built functions of OpenCV.

Acknowledgments

1. Professor Bruce Maxwell : author of csv_util files.
2. Inbuilt Moment functions in open CV. //https://learnopencv.com/shape-matching-using-hu-moments-c-python/
3. Map data structure. https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/
4. Putting a text in image on OpenCv. https://www.geeksforgeeks.org/write-on-an-image-using-opencv-in-cpp/