Overview of the project:

In this project, we are building a Real time 2D-object recognition system. The final objective is that the system should identify a set of given objects placed on a white surface.

In the first task, we worked on thresholding the input video thereby separating foreground and background. In the next task, we cleaned up the obtained binary image using morphological filtering strategies. Following that, we segmented the cleaned image into regions using connected component analysis and region growing.

In the fourth task, we computed features for the major region in the image using moments and plotted the oriented bounding box. Further, we added a training mode where the user can collect the feature vectors of known objects and store them in a database. For the sixth task, we have used scaled Euclidean distance metric to classify new images. Following that, we have implemented KNN classifier to find the label when given a new image.

Further, we have evaluated the performance of our system by building a confusion matrix for all the labels in the dataset. We have also attached a demo video of our system working.

As part of this project, we have worked on the following extensions.

- **Extension 1:**

  Added more than 10 objects (17) to the DB.

- **Extension 2:**

  Segmentation algorithms from scratch. Apart from implementing the Thresholding and morphological operations like dilation and erosion from scratch.

- **Extension 3:**

  Implemented Otsu's algorithm from scratch to threshold the binary image.

- **Extension 4:**

  Enabled our system to identify multiple objects at once and also request for new label from the user if it is an unknown object.

1. **Threshold the input video**

We have implemented **thresholding from scratch**. we have implemented thresholding in gray scale. Through our experimentation we have found that threshold value of **124** is giving better results. We are using our own gray scale implementation.

Here are the weights we are using,

Channel/Color	Weight

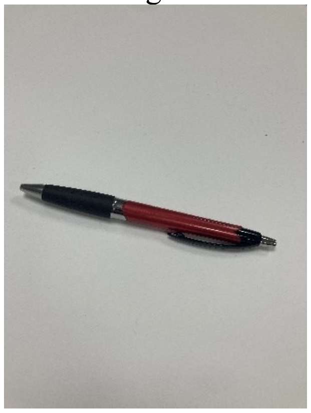
Blue	0.206
Green	0.588
Red	0.206

We have also experimented by converting the image into HSV color space and gray scale. The thresholding in gray scale was better than that of in HSV color space on the set of images we were training.

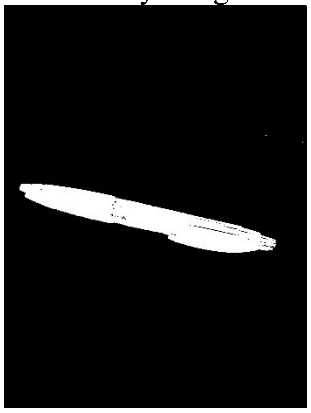
Further, we have tried implementing median and gaussian blur before thresholding. However, we do not see any improvement in the performance for our set of images. We have also explored the K-means approach to threshold the image as well as, which also provided similar results. So, we decided to perform direct thresholding on the gray scale of the given input image.

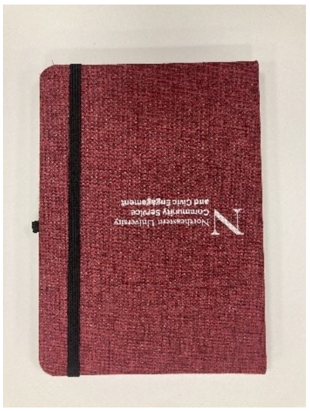
Additionally, we have implemented adaptive thresholding using Otsu's method as an Extension.

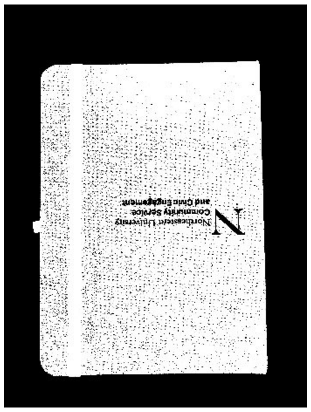
Original

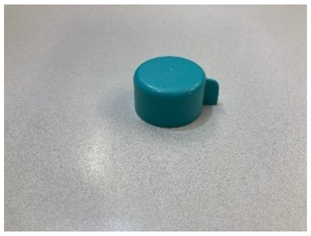
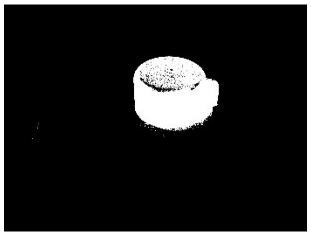


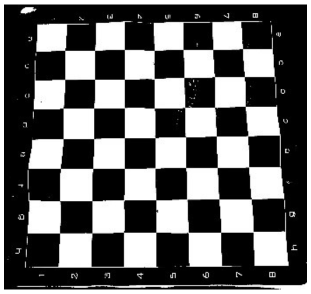
Binary Image

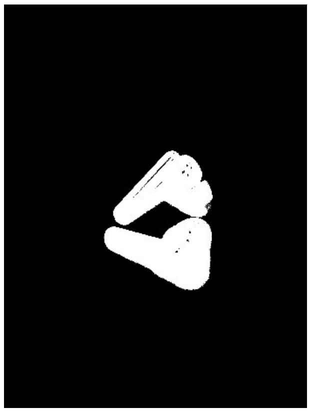








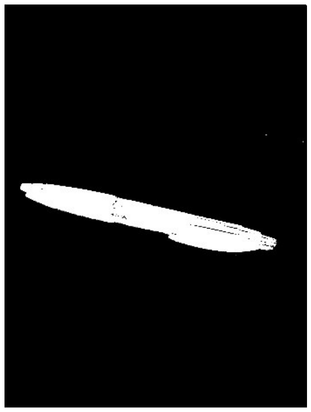
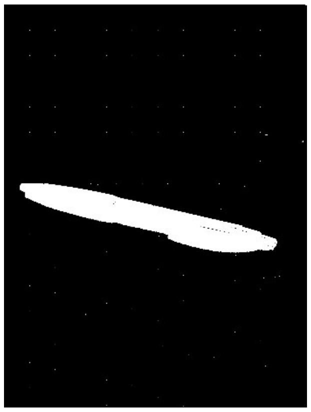
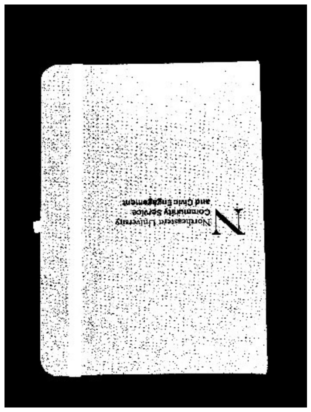
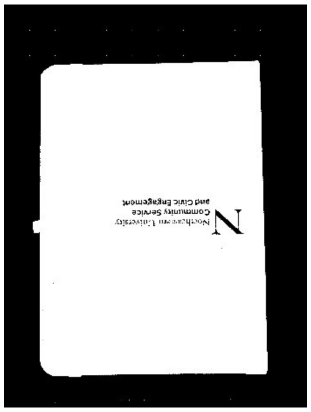
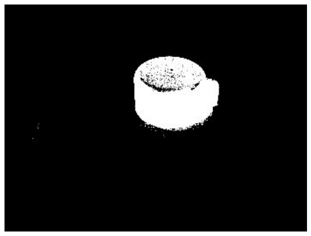
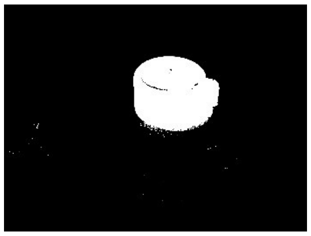
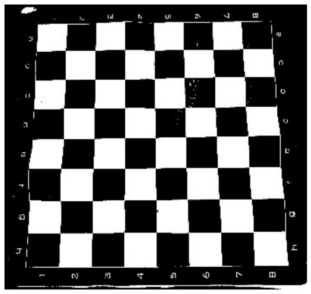
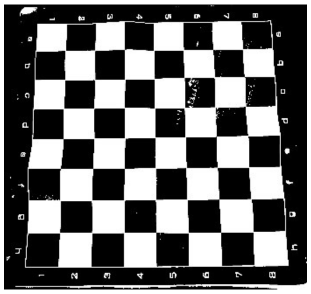


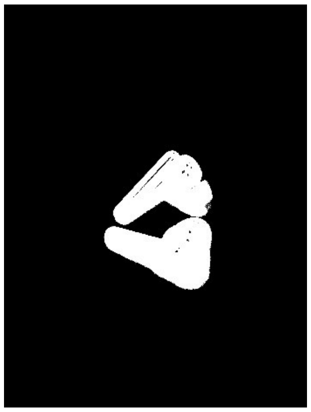
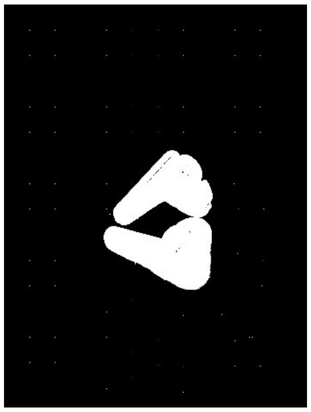
2. **Clean up the binary Image**

We have implemented **cleaning up the binary image from scratch.** we have first implemented the grassfire algorithm to compute the distance transform for the threshold image.

We then performed four times five 8-connected dilations followed by 4-connected erosions. We have arrived at these parameters based on experimentation with our set of images. Intuition is that when capturing an image, we might have some glare captured at the portion of the image, which might be marked as background, despite being foreground pixel. So, we opted to do the dilation first with 8 connected technique to get the rid of these issues and then erode with 4 connected technique to get back to same size as we started.

Binary image Cleaned image

3. **Segment the image into regions.**

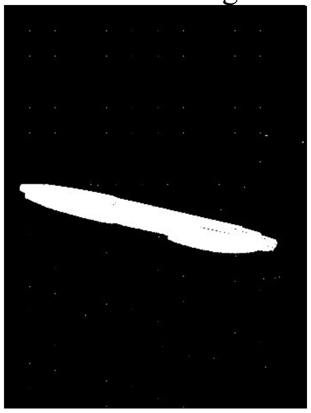
We have implemented **segmentation from scratch**. In the implementation, first we are computing the region map of the binary image (cleaned image) using connected component analysis/region growing.

We have implemented **both region growing**, connected component analysis using **union find**.

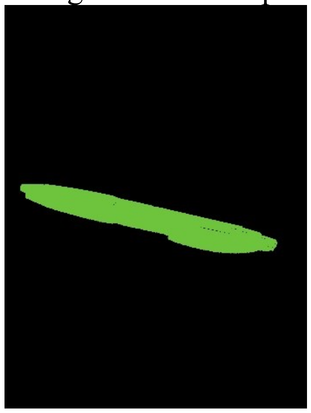
After computing the region map, we colored the regions in the foreground with different random colors based on their region ID from the region map.

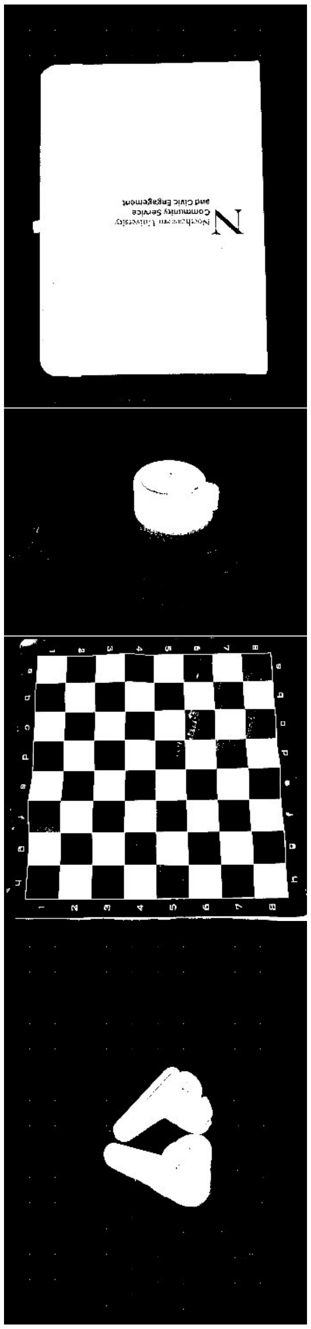
The results of segmentation are shown below,

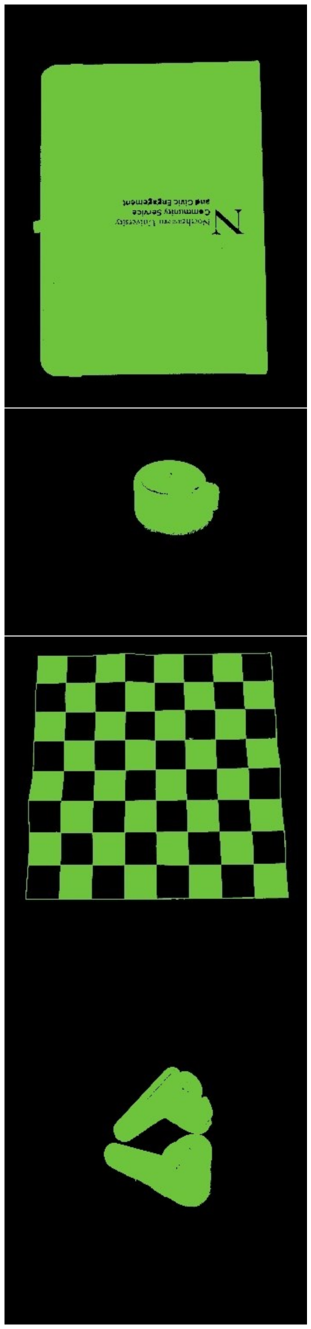
Cleaned Image



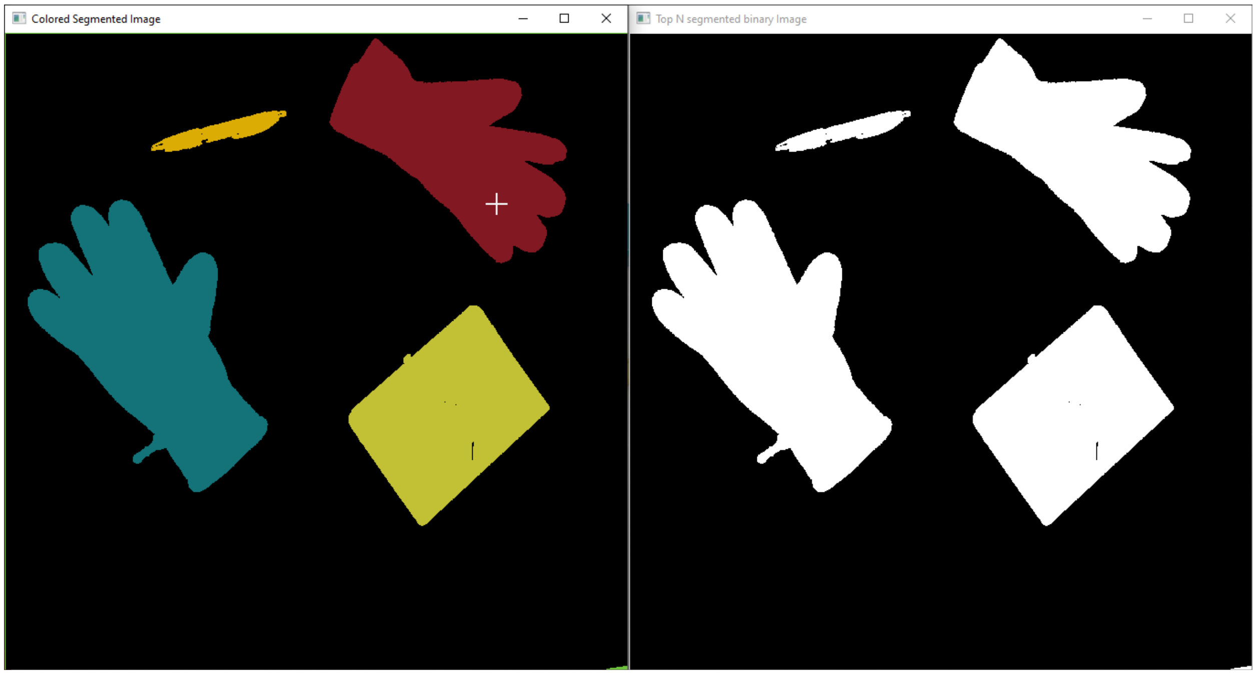
Segmentation Map







For each object you might observe same color as we have used Seed. This multiple color segmentation will be clear in the image with multiple objects in them. Like below:



4. **Compute features for each major region**

In this task, we have used OpenCV Moments function to compute the moments which are then used to compute the centroid, Angle of least central moment.

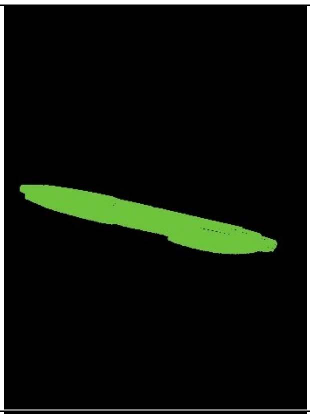
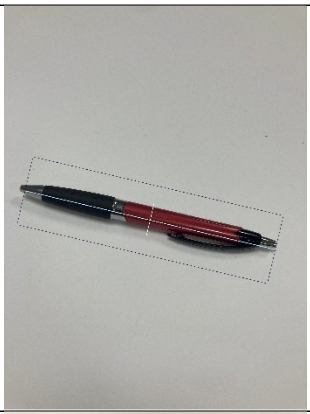
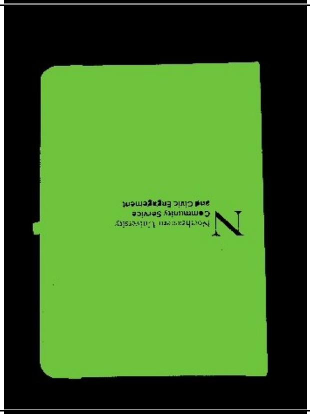
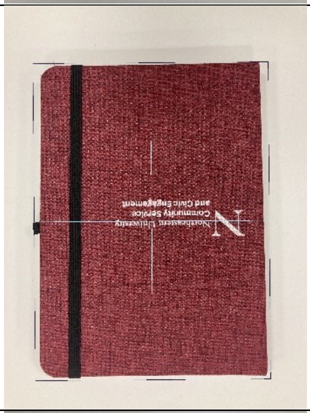
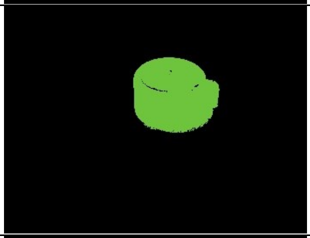
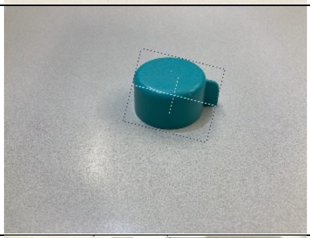
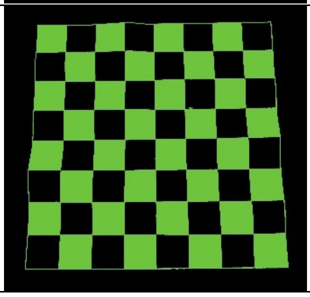
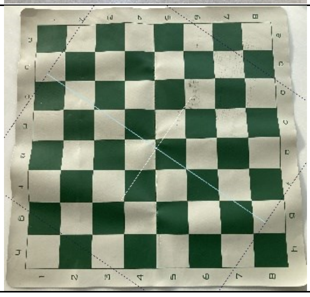
We are computing orientation axes and oriented bounding box without using any OpenCV functions except for calculating moments. We are using the centroid and the angle of least central moment to find the orientation axes. We are then finding the minimum Row, minimum Column, maximum Row, and maximum Column to compute width, height and then the oriented bounding box as discussed with professor.

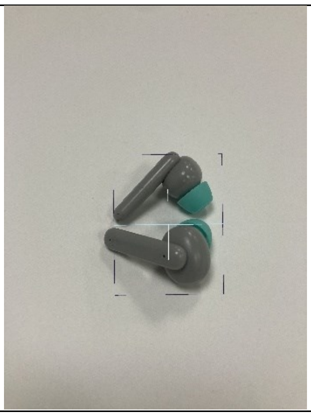
We have also computed hu moments using OpenCV huMoments function. These are **7** values obtained by using previous computed moments. huMoments function takes in moments as argument and produces **7** translation, scale, and rotation invariant values.

We are using the following as features in the feature Vector,

- Height/width ratio
- percent fill
- **7** huMoments

Segmented Image	Bounding Box

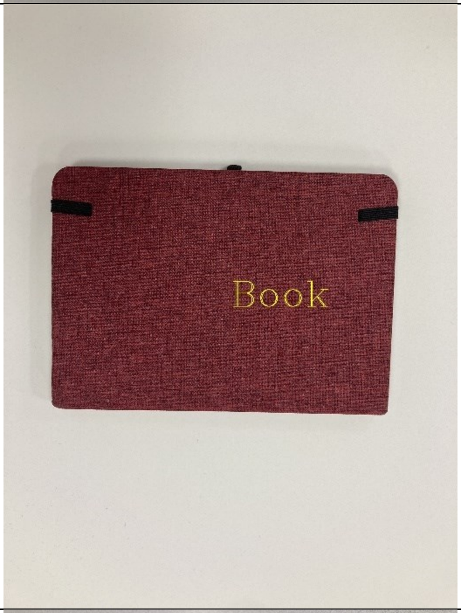
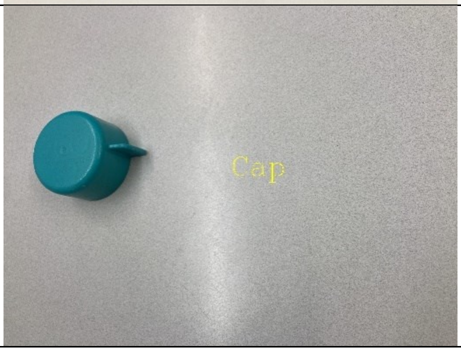
5. **Collect training data**

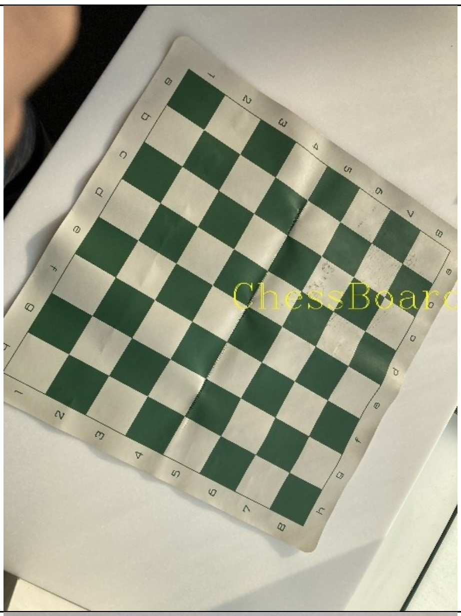
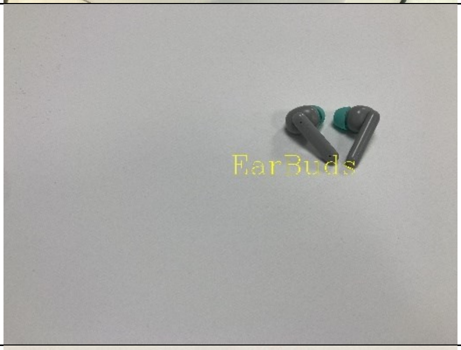
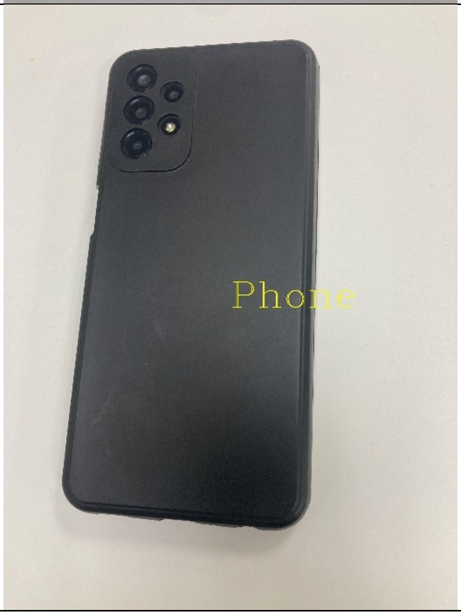
We have implemented our training system in such a way that there will be two modes available for the user, manual and automatic mode. In the manual mode, the user will be shown an image and will be prompted to enter the label for each image. Whereas in the automatic mode, the labels will be given based on the filename of the image. By default, the train runs in automatic mode.

Our system iterates over the database of training images folder and for each image file, it calculates feature vector. Depending up on the mode, the label is automatically fetched from the name of the file or is requested for the user to enter the valid label (cannot be empty) by displaying the file read to the user. After obtaining the label, this information along with feature vector calculated for the image is stored in a csv file (features.csv), so that during the object recognition process, we iterate only over the features.csv file to calculate the distance of the target image from the images in DB to predict the most likely label for the image.

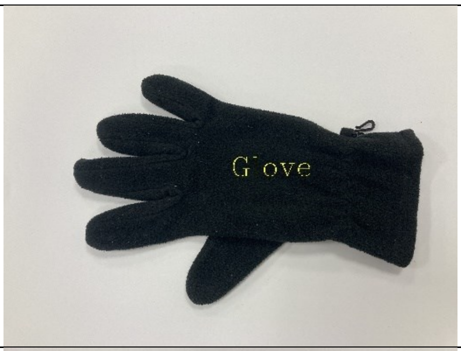
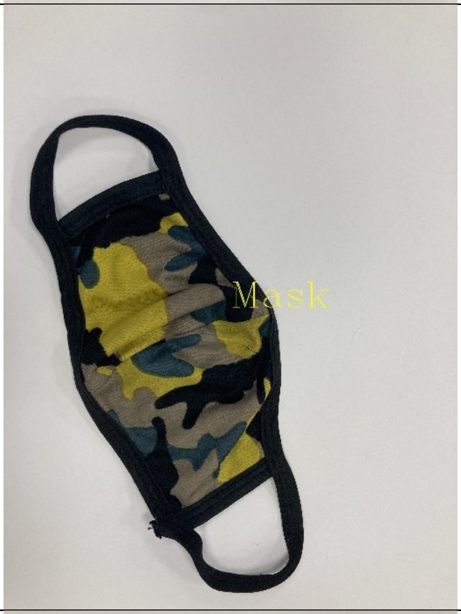
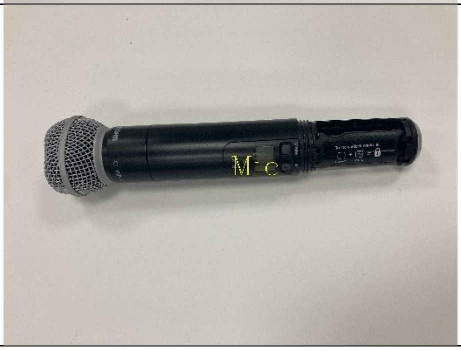
6. **Classify new images.**

We are using scaled Euclidean distance metric to obtain the closest label based on the features in the .csv file (features.csv file).

Pen
Book
Cap

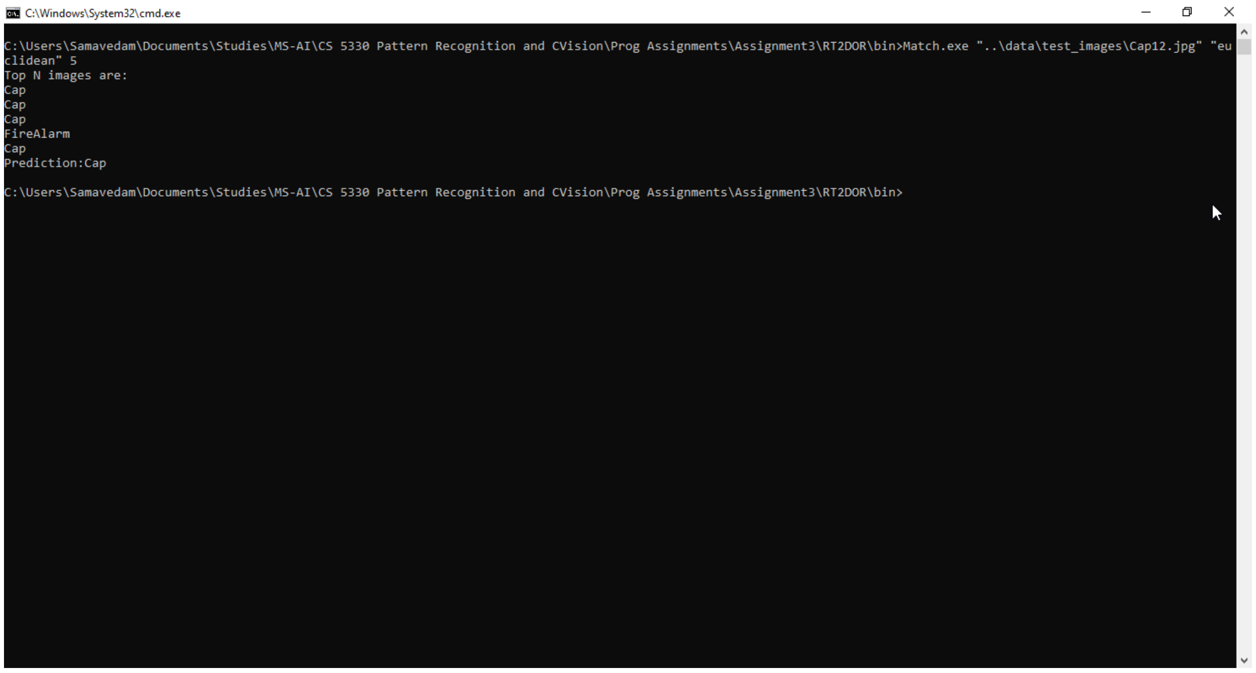
ChessBoard
EarBuds
Phone

Beanie
FireAlarm
GlassesCase

	Glove	
	Mask	
	Mic	

7. **Implement a different classifier**

We have implemented K-Nearest Neighbor classifier with K =5. We have used nearly 10-20 training examples for each image. Based on the distance calculated, we are attaching the label which is most occurring in the labels predicted. In case of conflict in the choice of labels (say closest images have labels “Book”, “Phone” twice), we have taken into the closest distance as the primary/target label.

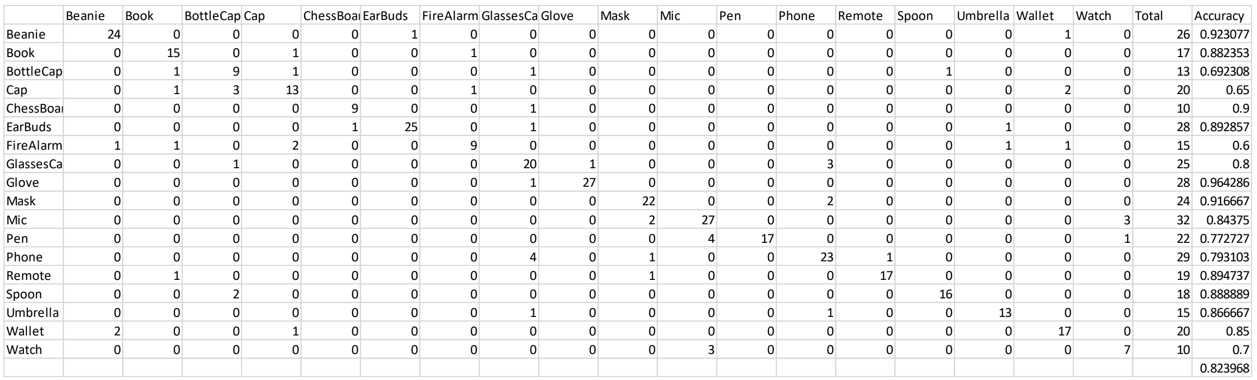


```
C:\Users\Samavedam\Documents\Studies\MS-AI\CS 5330 Pattern Recognition and CVision\Prog Assignments\Assignment3\RT2DOR\bin>Match.exe "..\data\test_images\Cap12.jpg" "eu
clidean" 5
Top N images are:
Cap
Cap
Cap
FireAlarm
Cap
Prediction:Cap

C:\Users\Samavedam\Documents\Studies\MS-AI\CS 5330 Pattern Recognition and CVision\Prog Assignments\Assignment3\RT2DOR\bin>
```

8. **Evaluate the performance of your system**

We have written code to loop through the test images directory to predict the label based on the chosen technique and then cross validate it with the ground truth (label) fetched and store this information into a confusion matrix file. The confusion matrix for our system for the labels is as below.



	Beanie	Book	BottleCap	Cap	ChessBoai	EarBuds	FireAlarm	GlassesCa	Glove	Mask	Mic	Pen	Phone	Remote	Spoon	Umbrella	Wallet	Watch	Total	Accuracy
Beanie	24	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	26	0.923077
Book	0	15	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	17	0.882353
BottleCap	0	1	9	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	13	0.692308
Cap	0	1	3	13	0	0	1	0	0	0	0	0	0	0	0	0	2	0	20	0.65
ChessBoai	0	0	0	0	9	0	0	1	0	0	0	0	0	0	0	0	0	0	10	0.9
EarBuds	0	0	0	0	1	25	0	1	0	0	0	0	0	0	0	1	0	0	28	0.892857
FireAlarm	1	1	0	2	0	0	9	0	0	0	0	0	0	0	0	1	1	0	15	0.6
GlassesCa	0	0	1	0	0	0	0	20	1	0	0	0	3	0	0	0	0	0	25	0.8
Glove	0	0	0	0	0	0	0	1	27	0	0	0	0	0	0	0	0	0	28	0.964286
Mask	0	0	0	0	0	0	0	0	0	22	0	0	2	0	0	0	0	0	24	0.916667
Mic	0	0	0	0	0	0	0	0	0	2	27	0	0	0	0	0	0	3	32	0.84375
Pen	0	0	0	0	0	0	0	0	0	0	4	17	0	0	0	0	0	1	22	0.772727
Phone	0	0	0	0	0	0	0	4	0	1	0	0	23	1	0	0	0	0	29	0.793103
Remote	0	1	0	0	0	0	0	0	0	1	0	0	0	17	0	0	0	0	19	0.894737
Spoon	0	0	2	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	18	0.888889
Umbrella	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	13	0	0	15	0.866667
Wallet	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	17	0	20	0.85
Watch	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	7	10	0.7
																				0.823968

Average accuracy of our system is around 83%.

9. **Capture a demo of your system working**

https://youtu.be/ZORyRWK4H6c

Extensions:

1. We have extended our model to recognize additionally 7 objects apart from the required 10 objects, making our model predict for nearly 17 objects.
2. We have Segmentation algorithms region growing and union find algorithms from scratch. Apart from implementing the Thresholding and morphological operations like dilation and erosion from scratch which internally use the grassfire transform implemented from scratch.
3. Implemented Otsu's algorithm from scratch to threshold the binary image. Based on the readings provided by the professor about the project, by going through the Shapiro book, we have implemented the Ostu's algorithm.

4. Enabled our system to identify multiple objects at once and also request for new label from the user if it is an unknown object. Currently this is limited to static images only.

As chess pieces are untrained objects, they are labelled as unknown. We have achieved this by comparing the existing label's distance from the target image and by thresholding it over a certain threshold. We then ask the user to enter new label and append the feature vector along with the label to the features.csv file for future reference.

Reflections:

This project provided an overview of the morphological operations, their significance and how we can use the moments/ properties of the shape of the objects to recognize the objects.

Acknowledgements

Professor Bruce helped us to debug the connected component analysis implementation using union-find. We also had a discussion with Professor Bruce, about better ways to compute the oriented bounding box.

https://learnopencv.com/otsu-thresholding-with-opencv/

https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

https://docs.opencv.org/3.4/d0/d49/tutorial_moments.html

https://learnopencv.com/shape-matching-using-hu-moments-c-python/