CS5550: PRCV
Project 3: Real-time 2-D Object Recognition Report

Tejaswini Dilip Deore

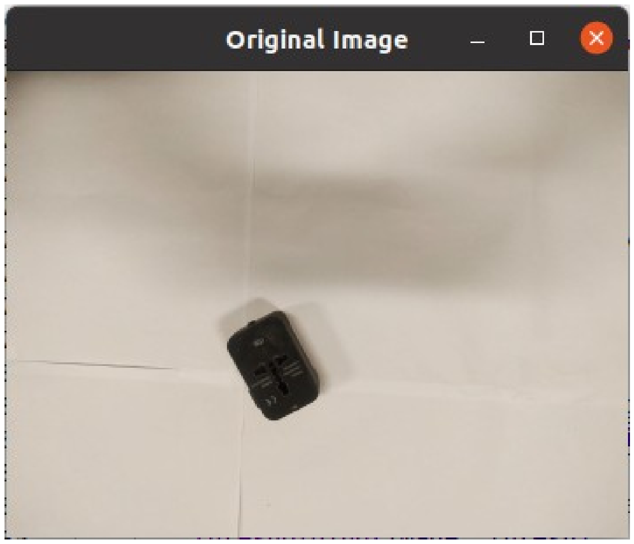
Project Description:

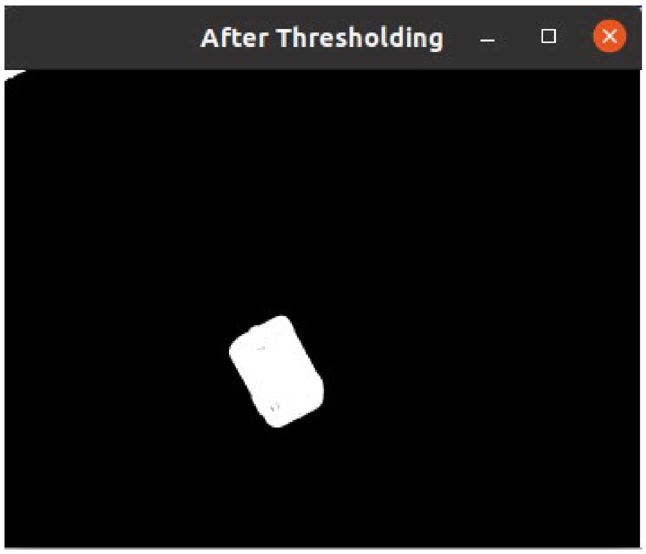
This project implements 2-D object recognition by identifying a specified set of objects placed on a white surface in a translation, scale, and rotation invariant manner from a camera looking down. This project uses an image directory of 11 different 2D objects of differentiable shape and a uniform dark color. Overall, the project converts input image to binary by thresholding, cleans up the image, gets region by image segmentation, calculates and stores features- HuMoments, width to height ratio, and % area filled, of the image. The program also trains the dataset with features for 11 objects and stores it as .csv file. This trained dataset is then used to classify new images using two different classifiers.

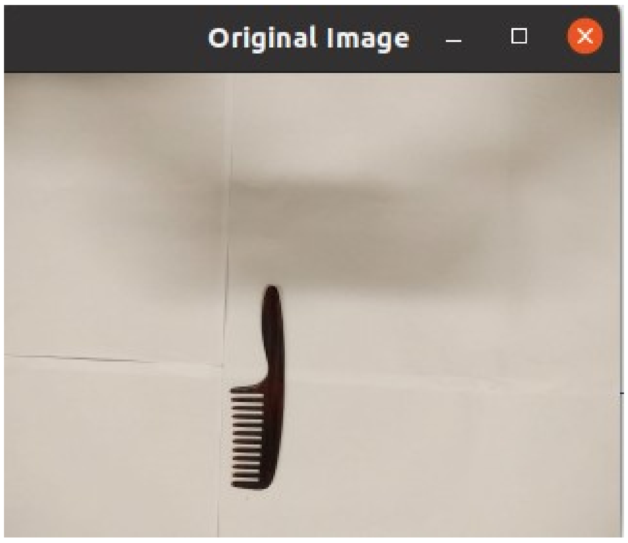
The objects recognized in this project are: Pen, Mask, Adapter, Watch, Calculator, Comb, Earphones, ZEDBox, Cloth, BottleCap, SpectacleCase

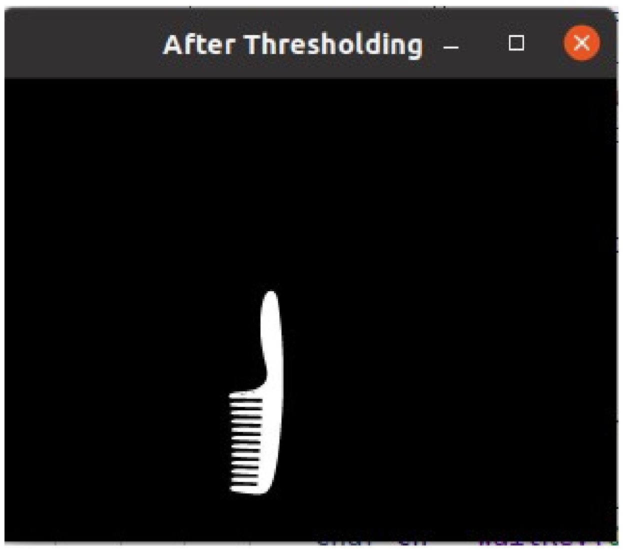
1. Threshold the input video:

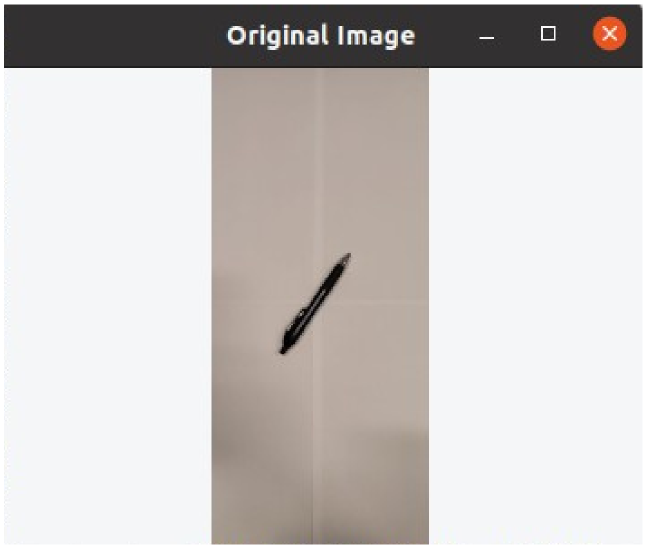
The input image is converted to binary using image thresholding. The image is pre-processed before thresholding to get rid of noise by blurring it using inbuilt function 3x3 GaussianBlur(). Then the pixel values below the defined threshold value are set as foreground (255) and rest as background (0). Following images show 3 examples of thresholded images.

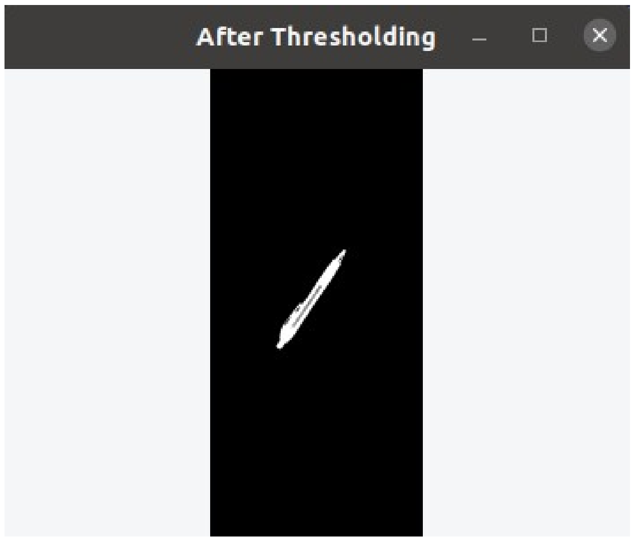






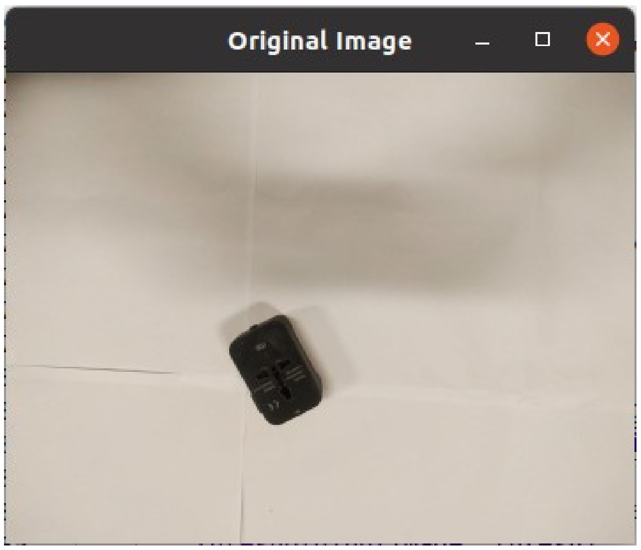


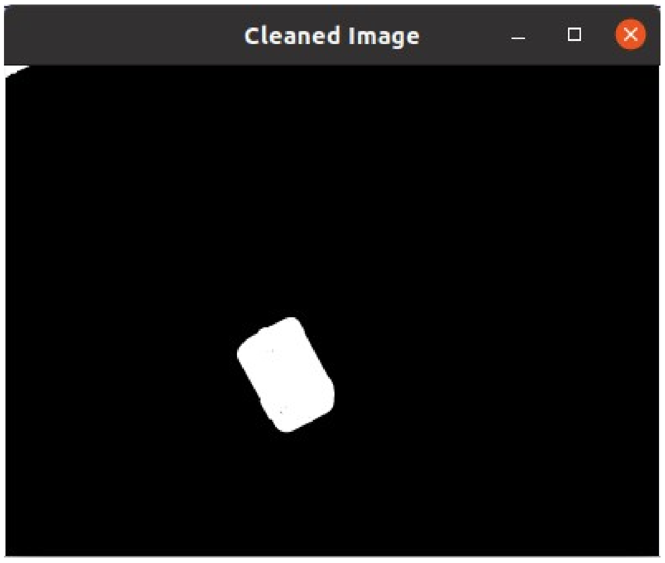


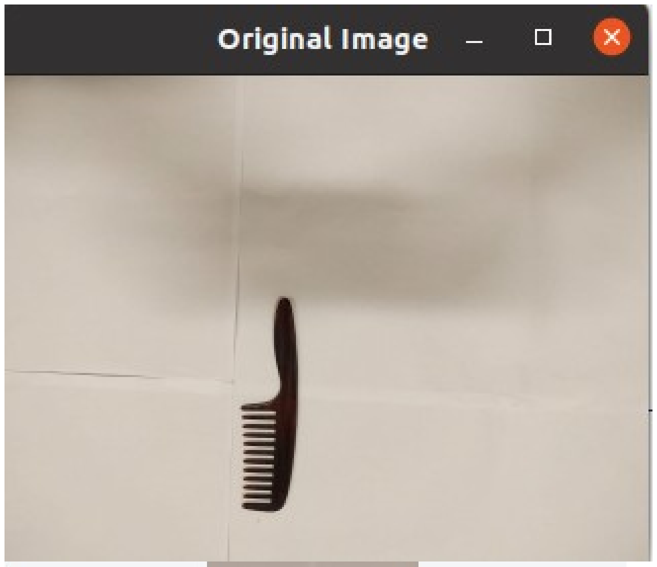


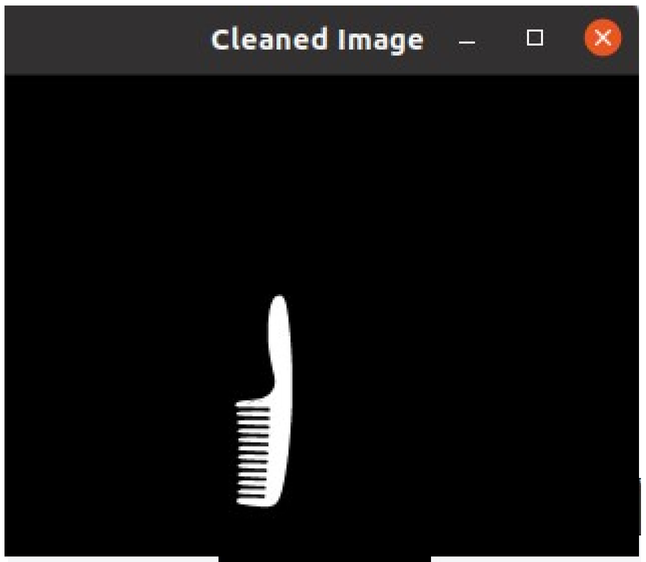
2. Clean up the binary image:

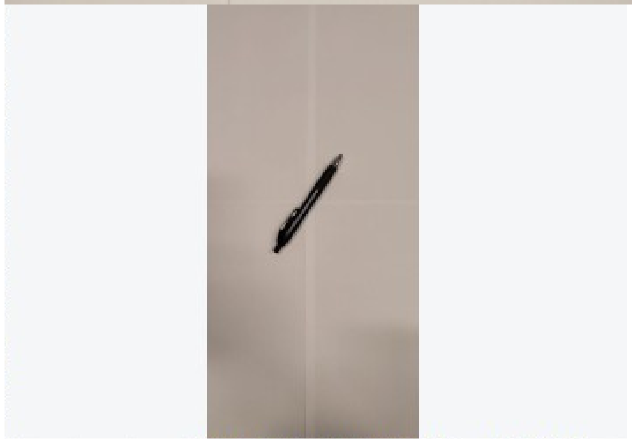
Thresholded image is cleaned up by applying morphological filters(4-connected erosion followed by 8-connected dilation) to get rid of noise and holes. I applied erosion first to get rid to small noise that was present in the image. Since the thresholded images had more holes compared to noise, I decided to use more iterations of 8-connected dilation by accessing each pixel and it's 8 neighbors. Fig below shows binary image before and after thresholding. Following images show 3 examples of morphologically filtered images.

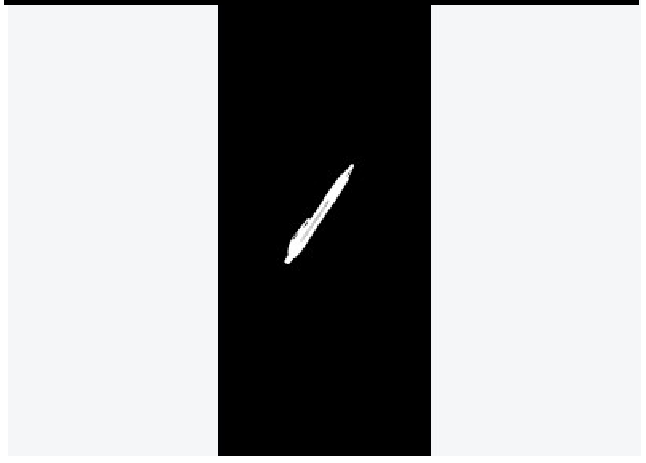






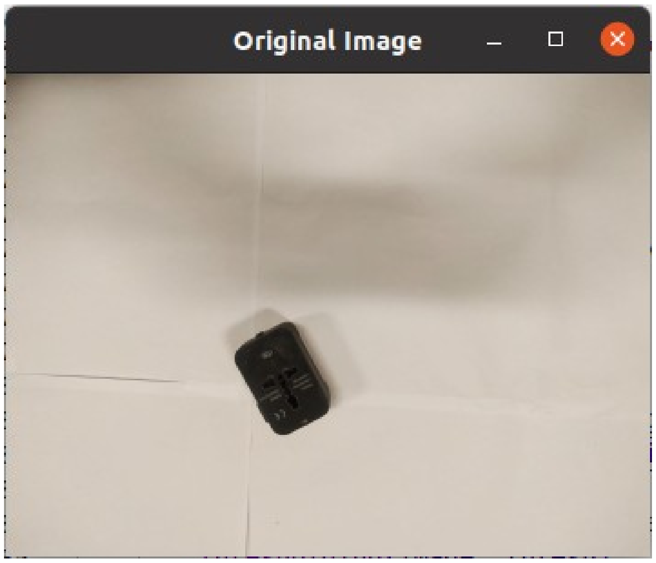


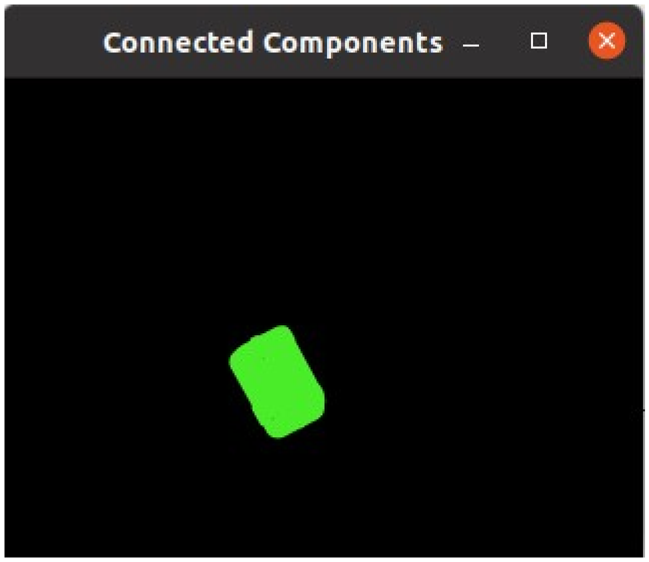


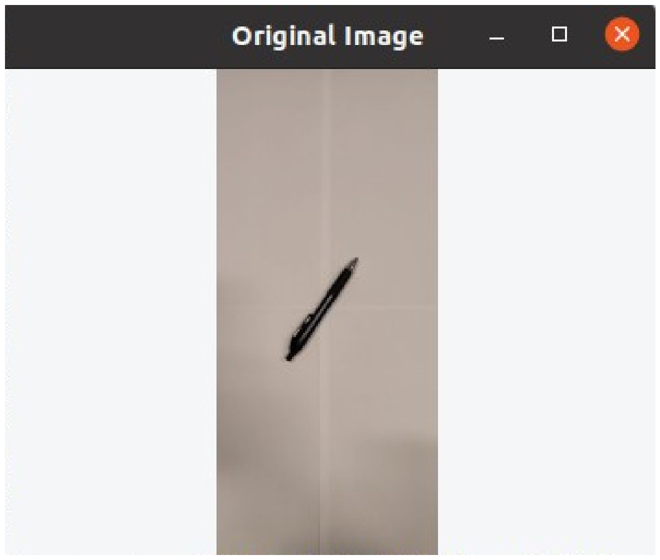


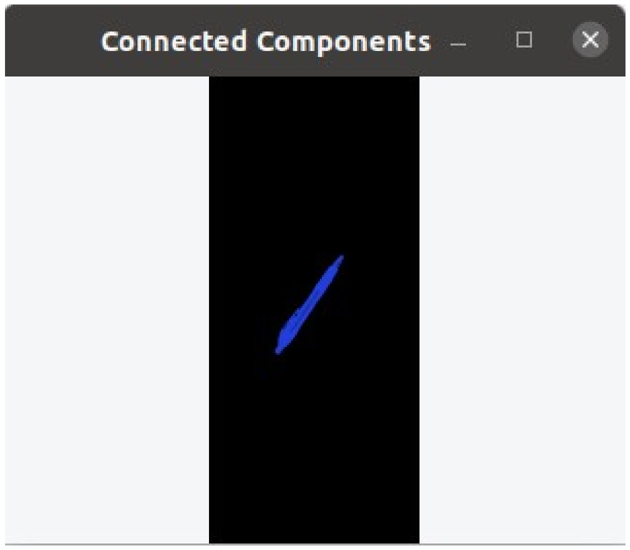
3. Segment the image into regions:

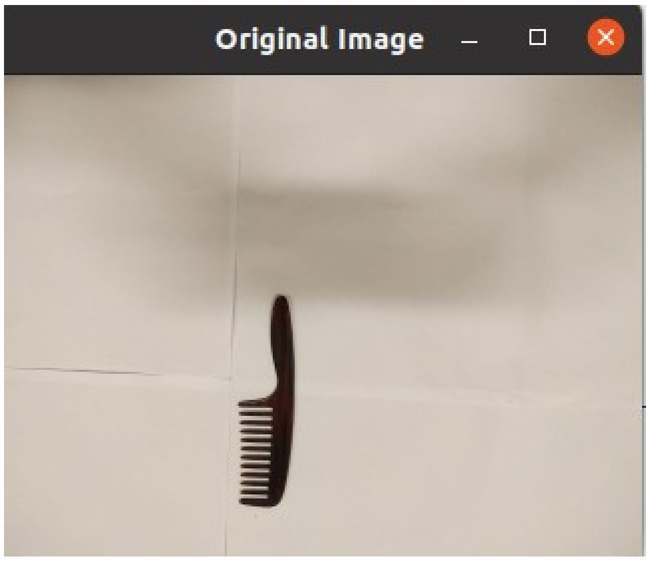
The OpenCV inbuilt function connectedComponentsWithStats() computes the connected components labeled image of binary image and also produces a statistics output for each label. The function segment_regions() ignores areas below given threshold size (in this case 5000) and recognizes largest 3 regions. Following images show 3 examples of segmented regions.

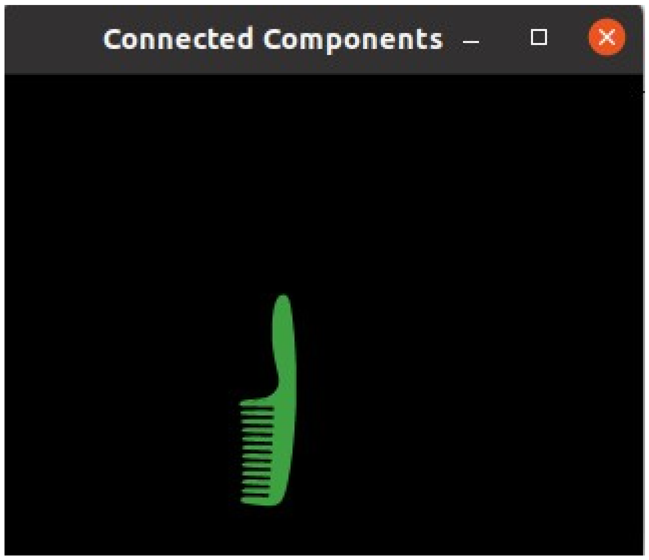










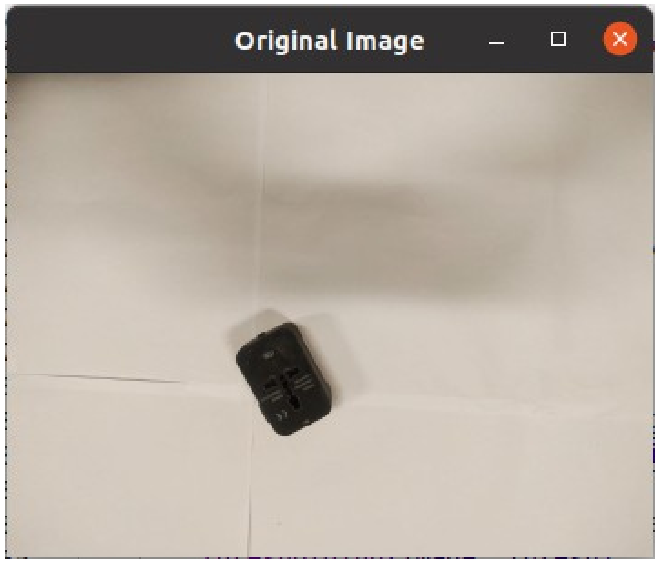


4. Compute features for each major region:

The features for region in this project are HuMoments, width to height ratio, and % area filled all these features are translation, scale, and rotation invariant. I have used OpenCV built-in functions moments() and HuMoments() to compute features for each major region. The function also draws bounding box around the region, and arrow in the direction of moment is also indicated.

The OpenCV moments() function calculates various moments of a given image, such as the centroid, area, and spatial moments. The statistical moments calculated by the moments() function in OpenCV are not inherently invariant to translation, scale, or rotation. Hence I used HuMoments() function in OpenCV that calculates a set of seven moments that are invariant to translation, rotation, and scale. The seven Hu Moments are represented by the variables hu1, hu2, hu3, hu4, hu5, hu6, and hu7. These variables are calculated using the central moments of the image and a set of complex mathematical equations. The resulting Hu Moments are invariant to translation, scale, and rotation, which means they can be used to identify objects regardless of their position, size, or orientation in the image. HuMoments hu5, hu6 and hu7 are are based on higher-order moments that are calculated from a smaller region of the image. In some cases, this can lead to numerical instability and inaccuracies in the calculated values of these moments. For this reason, I have used hu1-4 as features.

To create the rotated bounding box, the centroid and alpha (the angle between the x-axis and the least central x-axis) are utilized to construct the new x-axis and y-axis. Then each point in the initial coordinate system is projected onto the new coordinate system. Following images show 3 examples of regions showing the axis of least central moment and the oriented bounding box .


Original Image

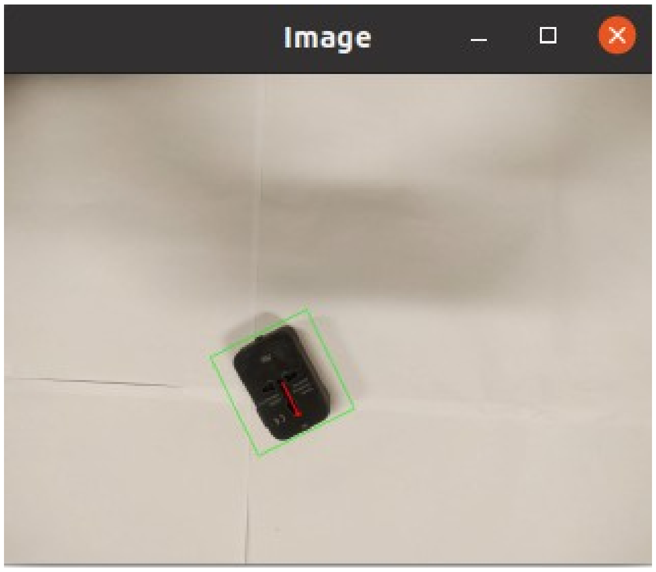
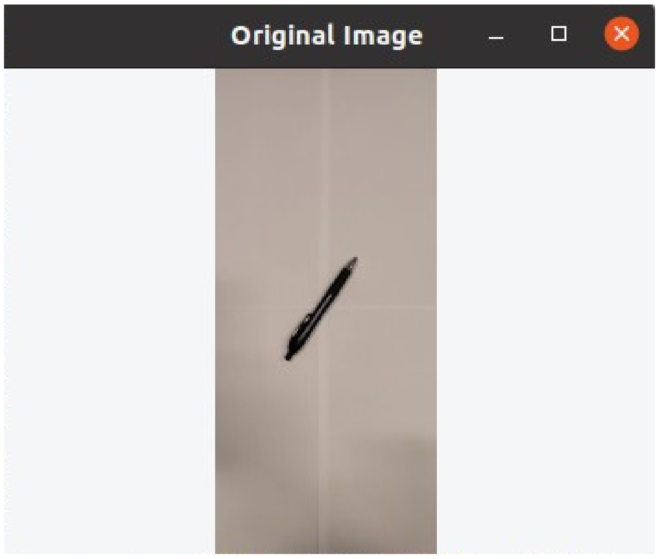



Image

Original Image

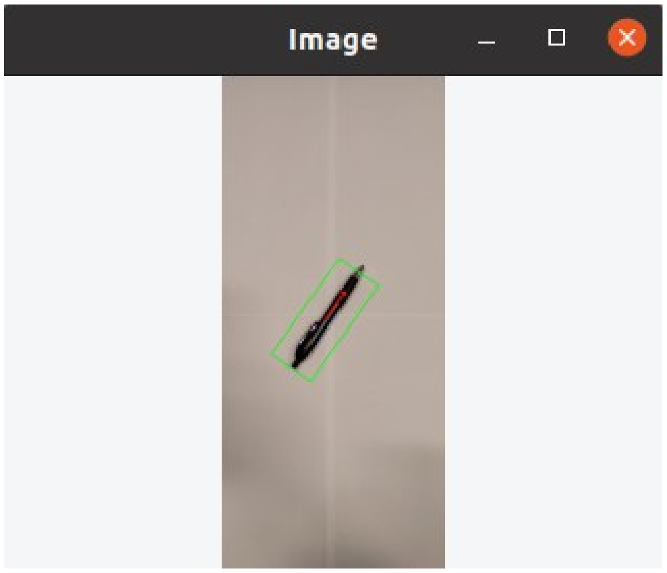
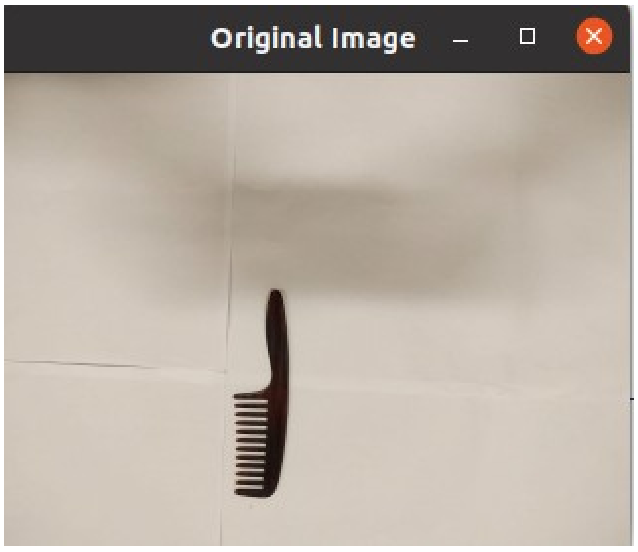
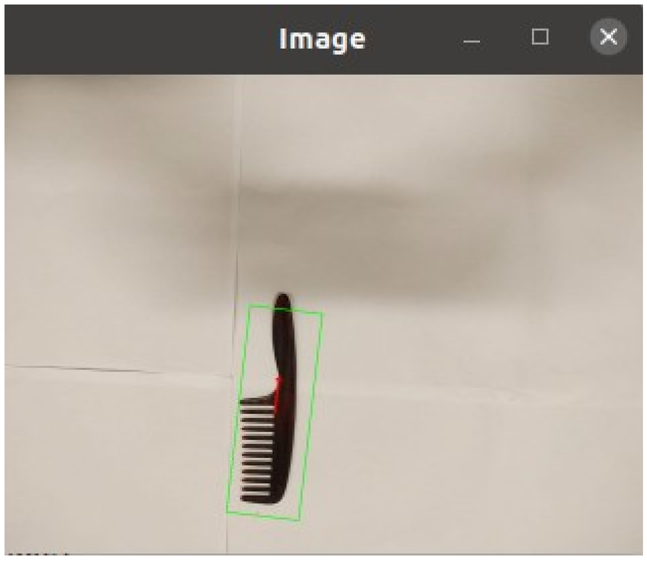



Image

Original Image

Image

5. Collect training data:

If the user enters 3rd command line argument as ‘t’, system will enter into training mode and will calculate feature vectors for all the images present in the training database path provided by the user. It displays a menu with known object set and asks user to enter the label using the keypress displayed in menu. Then these labels are stored in a .csv file along with corresponding feature vector. The training dataset is trained using 3 images of each object.
Click here to view sample video implementation of training few objects.

6. Classify new images:

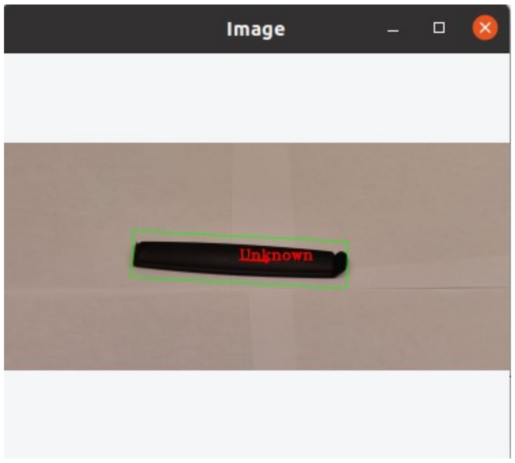
If the user enters 3rd command line argument as ‘c’, system will enter into classification mode and will displays the predicted label for the object in classification database using the feature vector data of trained images. This function uses Cosine Similarity as distance metric which compares how similar two feature vectors are. An unknown object is labelled as ‘Unknown’. This classification is implemented if the user enters ‘x’ as 4th command line argument.
Click here to view implementation of classifier.

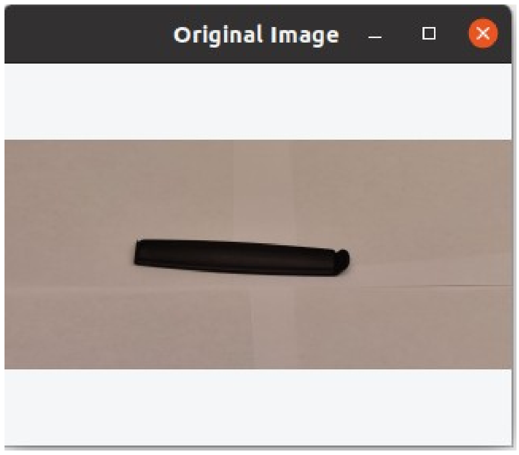
7. Implement a different classifier:

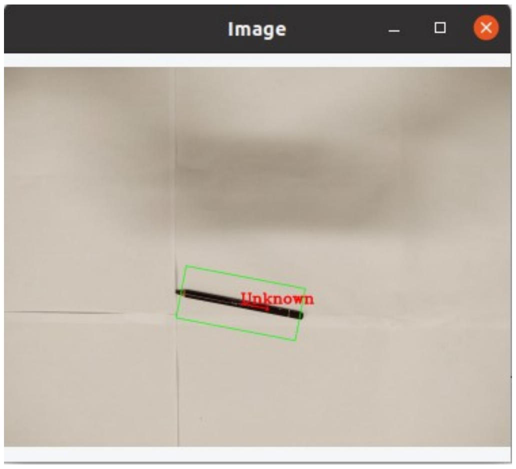
The second classifier implemented is **K-Nearest Neighbor matching** with k=3, which considers 3 nearest neighbors while predicting the label for image to be classified. This classification is implemented if the user enters ‘y’ as 4th command line argument.
Click here to view implementation of classifier.

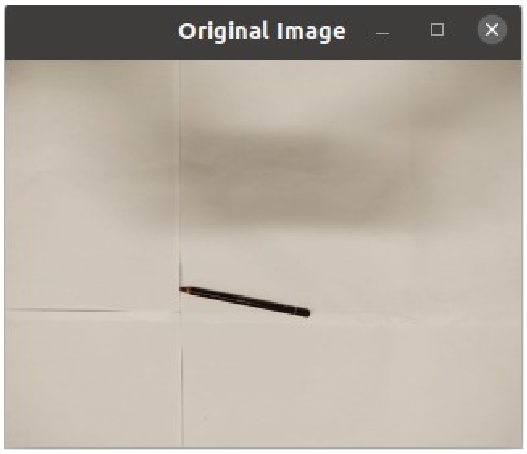
6. Extension:

- The system can recognize more than 10 objects
- An unknown object is labelled as ‘Unknown’ in task 6

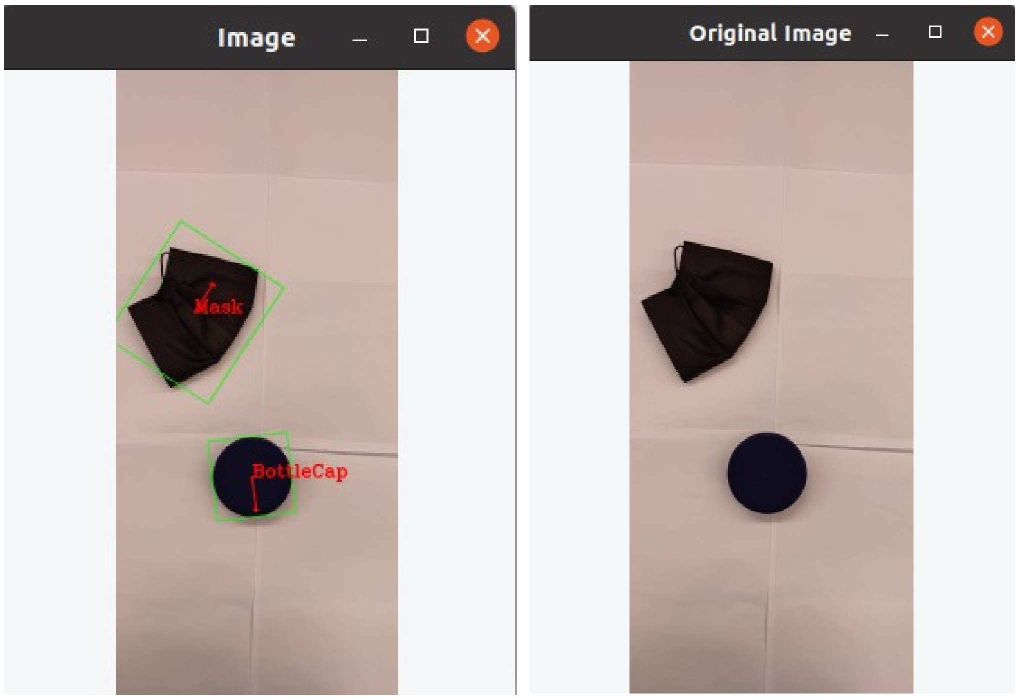








- The system can identify more that one object in task 6 and 7 (example shown in video)



7. Confusion Matrix:

Objects	Pen	Mask	Adapte r	Watc h	Calculato r	Comb	Earphones	ZEDBox	Cloth	BottleCap	SpectacleCase
Pen	5										
Mask		7									
Adapter			6								
Watch				6							
Calculator					6						
Comb						6					
Earphones							6				
ZEDBox								5			1
Cloth									6		
BottleCap										7	
SpectacleCase											6

Learnings from the project:

- Learned about 2D object recognition pipeline using OpneCV, with and without inbuilt functions
- Learned how to implement morphological filtering (erosion and dilation) from scratch
- Learned about different features of the images like moments, ration and % filled
- Learned about how to use drawing functions from OpneCV

Acknowledgement:

- https://docs.opencv.org/3.4/index.html
- https://docs.opencv.org/3.4/d6/d6e/group__imgproc__draw.html#ga07d2f74cadcf8e305e810ce8eed13bc9
- https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga107a78bf7cd25dec05fb4dfc5c9e765f
- https://cppsecrets.com/users/2025121111103101115104991049711010011497107971101161045556576410310997105108469911109/C00-OpenCVsortIdx.php