Real-time 2-D Object Recognition

Rahul Kumar
Northeastern University

1. SUMMARY

The purpose of this project is to implement object recognition of 2-D objects in real-time. It is based on the comparison of scale, translation, and rotation invariant features of a target object with labeled objects in a known database and assigning the label using a classifier system. The detection is done in real-time and to obtain the feature vector, the processing is done on video such as thresholding, cleanup, segmentation, and then feature detection. To label the object, two classifiers are implemented namely Nearest Neighbor and K-Nearest Neighbor, and distance is calculated using the Scaled Euclidean distance metric. This implementation is successful in recognizing 15 different objects in real-time. Also, any unknown objects (outside of the trained database) can be added to the database and can be learned in real-time.

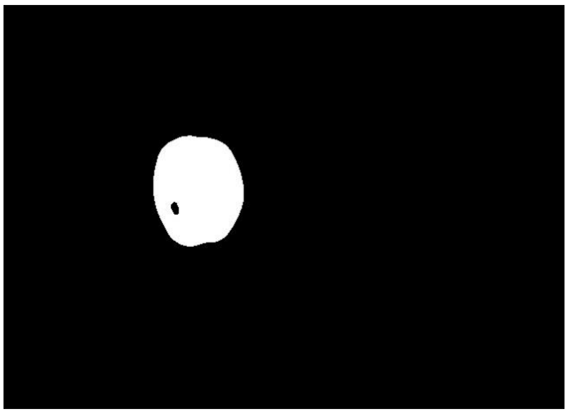
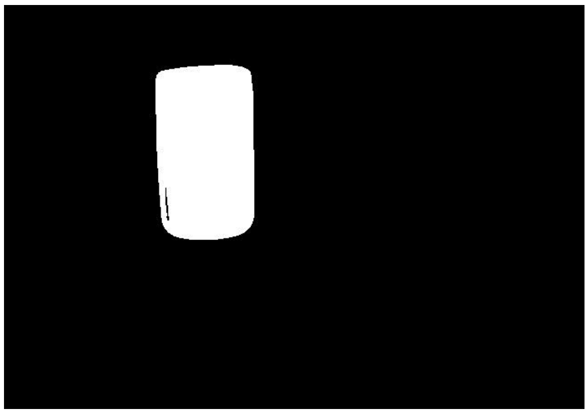
2. SETUP

To implement this project, setup is essential to get good results. The background is made of white and darker objects (to be recognized) and is kept in as foreground so the thresholding can be applied easily and the video is streamed through the Phone camera which is kept in a downward-facing position to the object. Also, the lightning in the test area is kept the same throughout the whole trial and testing period.

3. 2-D OBJECT RECOGNITION IMPLEMENTATION

3.1. THRESHOLDING

The first task in recognition is to separate the foreground from the background and this is achieved using thresholding the video stream. Before applying the threshold, the image is converted into grayscale and the same is blurred to make the area more uniform. Then histogram is plotted to observe the optimal pixel boundary to separate foreground from background and the appropriate threshold is applied to get a binary image in which all foreground pixels are set to 255 (below threshold) and background pixels are set to 0 (above threshold) [1]. This code is written from scratch.

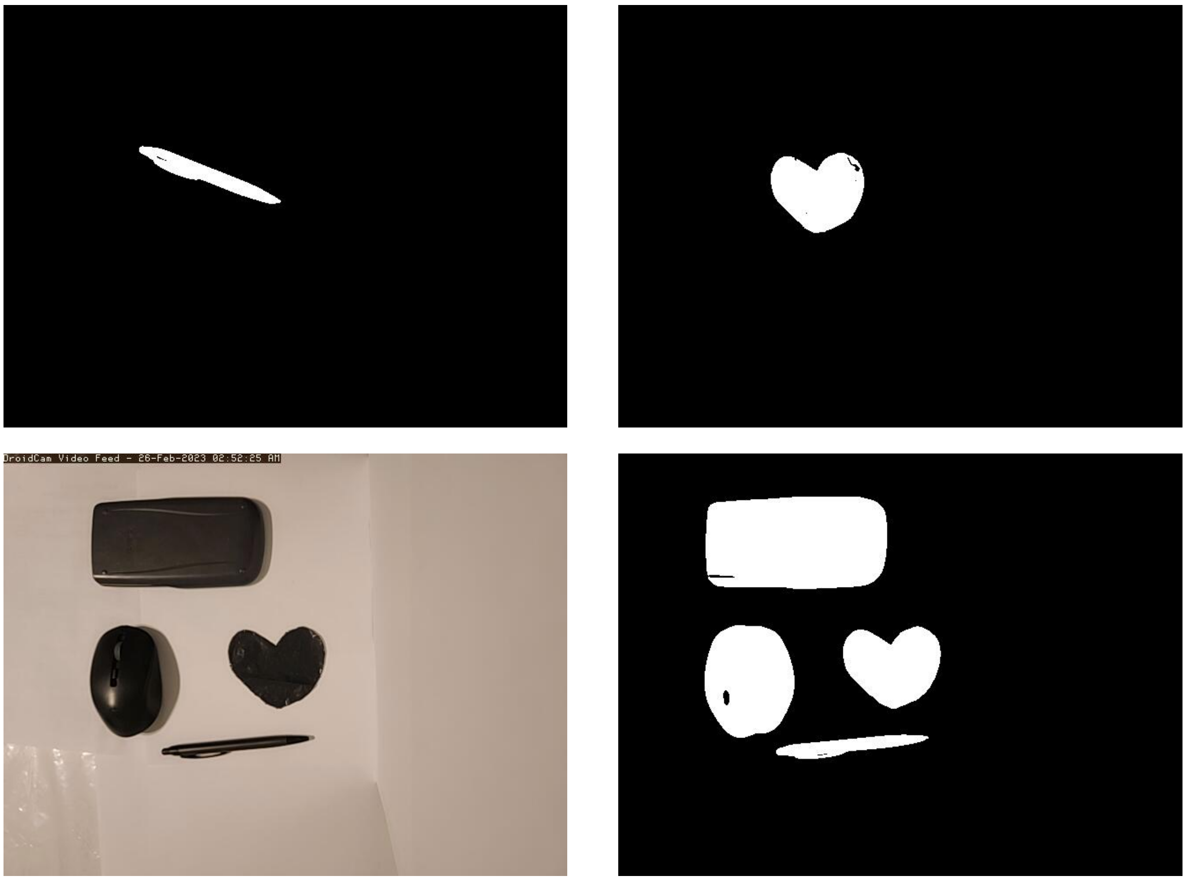


Fig. 1. Binary images showing the thresholded objects and the colored image is original

3.2. CLEANUP

A good threshold image with a white background can provide a very good result but still due to surface reflection, holes can appear on the foreground objects (can be seen in fig. 1). Before proceeding further, these holes to be filled, and spots, if any present, to be removed. To do so dilation and erosion are implemented [2]. Dilation filled the holes but it also increases the outer boundary, to retain the shape, erosion is also implemented. The cleaned-up image is free of holes and spots as shown below:

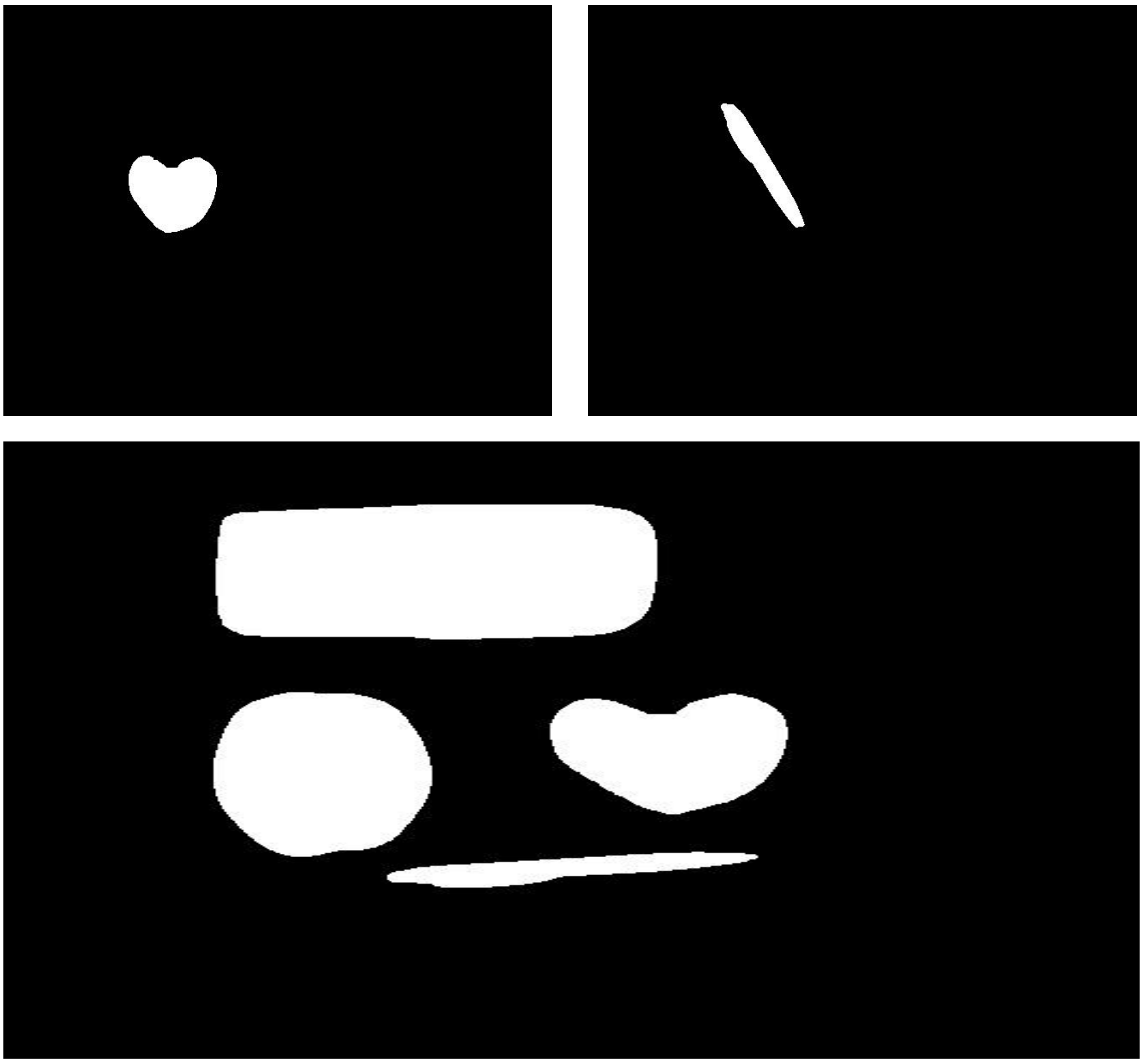


Fig. 2. Cleaned-up image after dilation and erosion on thresholded image

3.3. SEGMENTATION

Segmentation is done to find the regions of the foreground objects and generate a region map with region Id to segment the different regions. Connected component with stats, an OpenCV function [3] is used to achieve this. Also, an area threshold is applied on the segmented objects to let display only those areas that lie within a range and discard very small and very large areas. As an extension, multiple objects are also segmented at a time and different colors are given to different regions which lie in a given thresholded area.

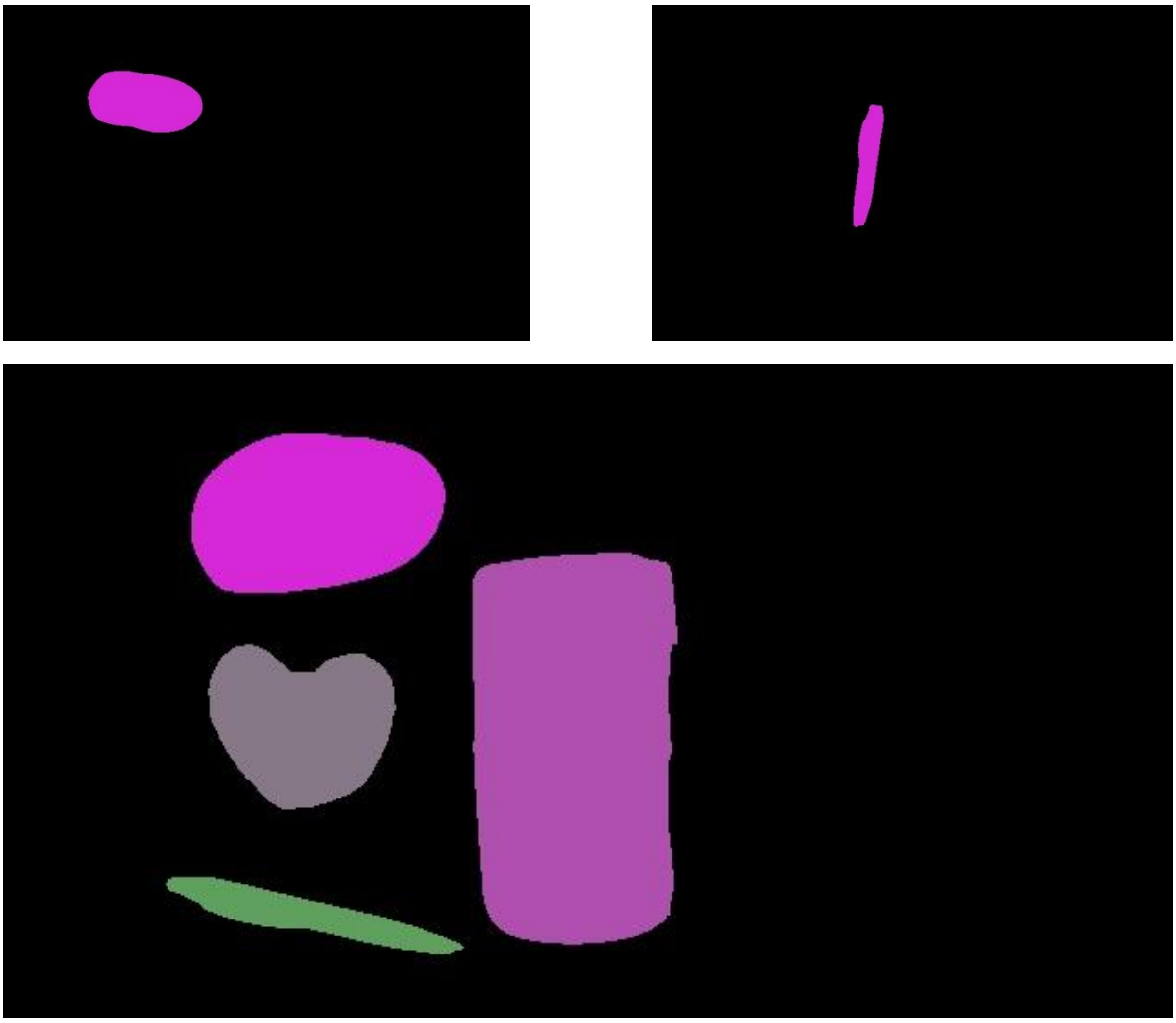


Fig. 3. Segmented image from cleaned-up image (showing different regions with a different colors)

3.4. FEATURE GENERATION

Given the segmented region and region Id from the segmented image, the feature vector is generated for each segmented region. Since the objects must be detected irrespective of position and orientation in the image, the features must be scale, translation, and rotation invariant. So, an axis of the least central moment is calculated and an oriented boundary box is drawn. Also, HuMoments are calculated using the OpenCV function, which is scale rotation and translation invariant [4]. To test this, one moment is displayed on the screen in real-time, and variation of this feature is observed by rotation and translating the object in the workspace. Each feature vector consists of 6 values, 4 are invariant moments, one of the rest is the area ratio of contour area and oriented boundary box area and the other one is the ratio of height to width.

As an extension, features along with the least central axis and oriented boundary box of multiple objects are displayed on the screen at a time. The below images show the implementation:

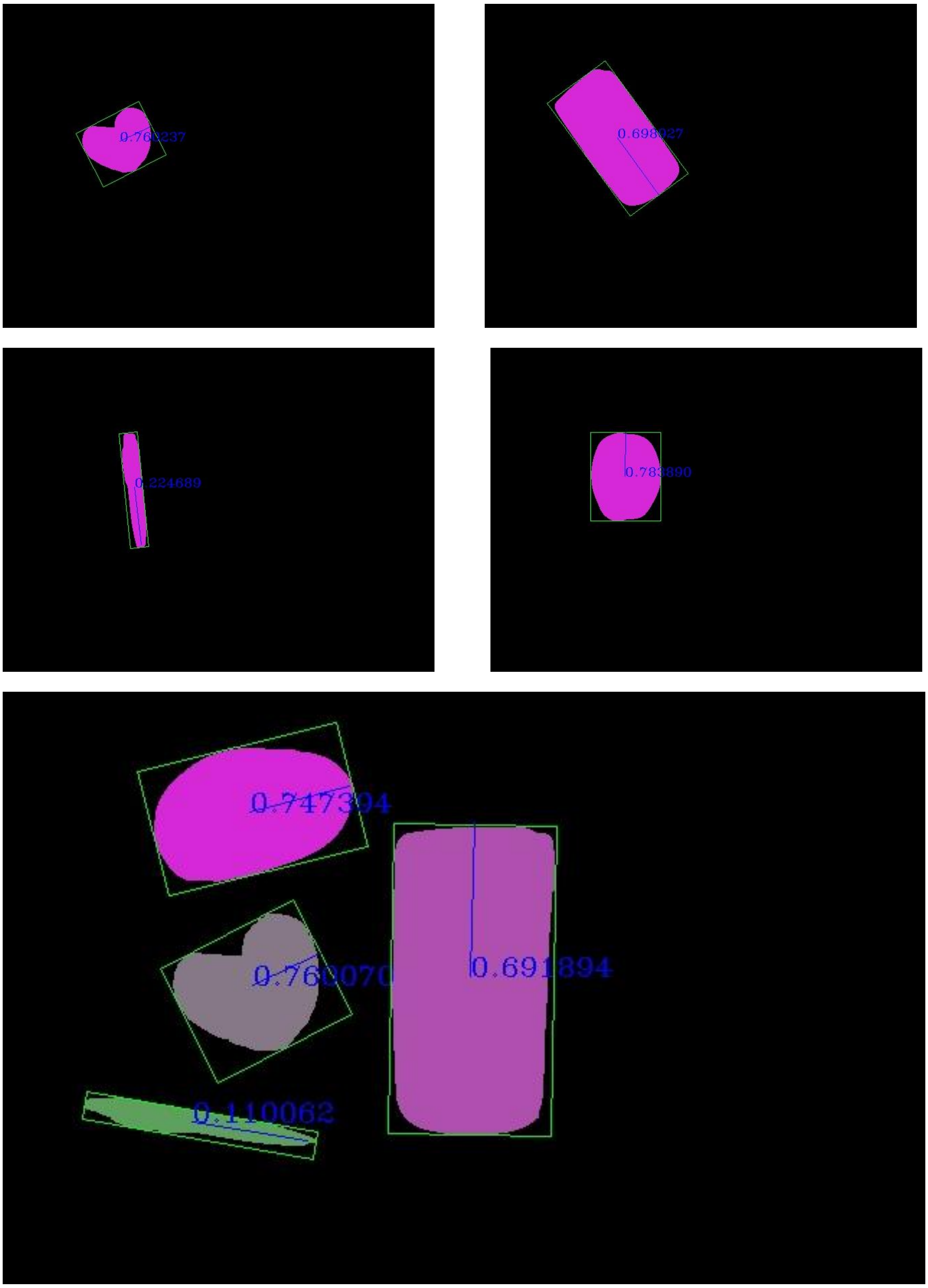


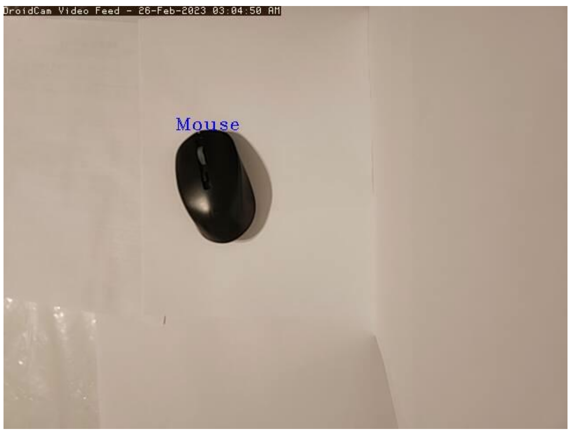
Fig. 4. Segmented image with feature vector along with least central axis and oriented boundary box for each region

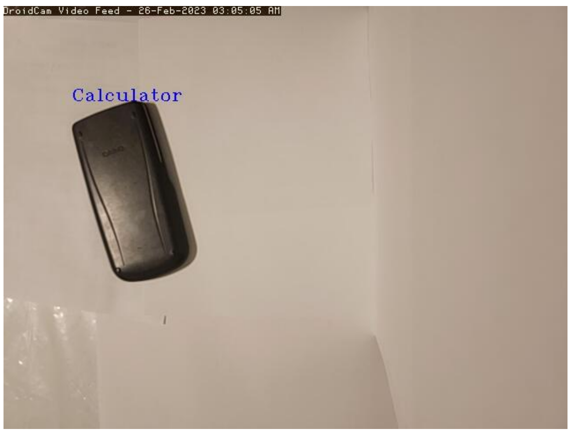
3.5. TRAINING DATABASE

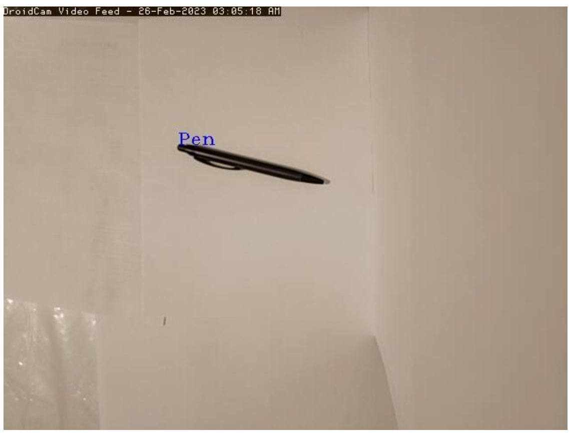
The training database is created using 15 objects with a minimum of 4 training sets for each class. It is implemented with key activation, when a user presses ‘n’, a prompt is displayed on the terminal asking for a label for a single object in the current video frame. When the label is entered, a CSV file is created and the label along with its feature vector is saved in the CSV file. To create multiple training examples of the same class, the same object is placed in different positions and orientations and its feature vector along with the label is appended in the CSV file.

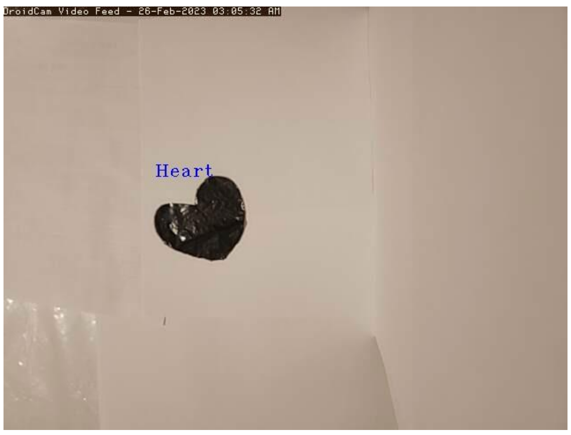
3.6. NEAREST NEIGHBOR CLASSIFICATION

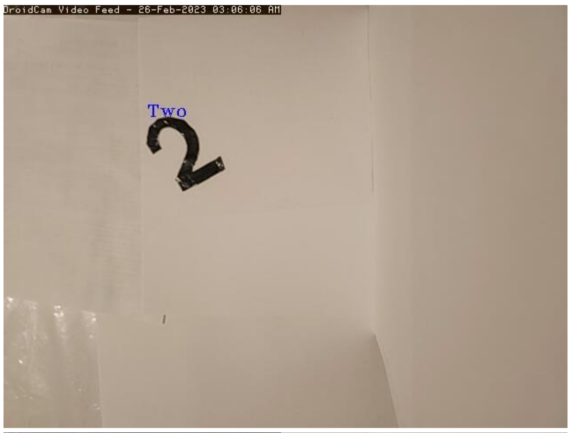
Objects are detected by implementing the Nearest neighbor classification which can be activated by pressing the key ‘c’. When the key is pressed, the feature vector of the target object is compared with the feature vector of labeled objects using the Scaled Euclidean distance metric, and label the object with the closest matching feature vector. The below images show the labeling of 15 different objects in real-time:










DroidCam Video Feed - 26-Feb-2023 03:06:06 AM
Two

DroidCam Video Feed - 26-Feb-2023 03:06:36 AM
Seven

DroidCam Video Feed - 26-Feb-2023 03:06:56 AM
CircularBase

DroidCam Video Feed - 26-Feb-2023 03:07:44 AM
Knife

DroidCam Video Feed - 26-Feb-2023 03:08:04 AM
capital_H

DroidCam Video Feed - 26-Feb-2023 03:08:41 AM
Diary

DroidCam Video Feed - 26-Feb-2023 03:08:56 AM
ShoePolish
Cherry

DroidCam Video Feed - 26-Feb-2023 03:09:09 AM
CardHolder
N
Northeastern University Library

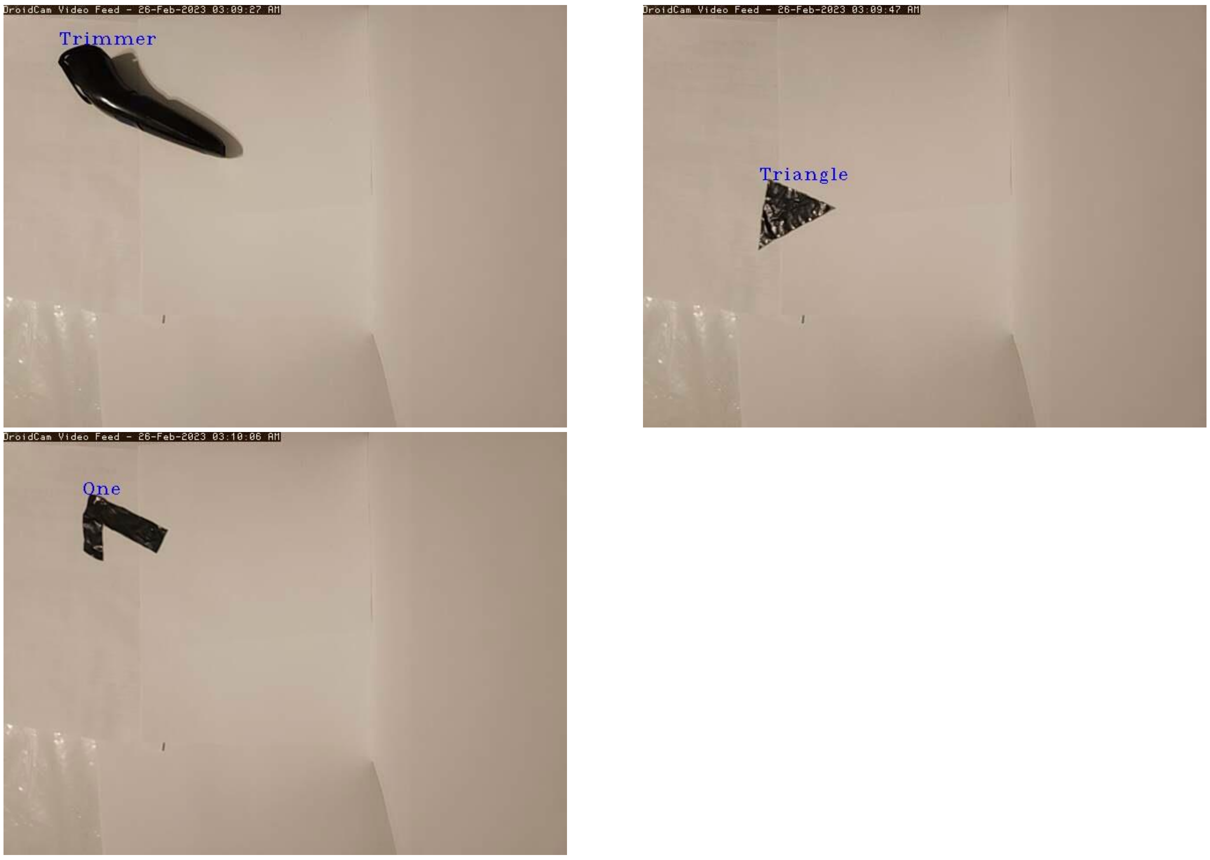

DroidCam Video Feed - 26-Feb-2023 03:09:27 AM
Trimmer
DroidCam Video Feed - 26-Feb-2023 03:09:47 AM
Triangle
DroidCam Video Feed - 26-Feb-2023 03:10:06 AM
One

Fig. 5. Labeling of objects in real-time

3.7. K-NEAREST NEIGHBOR CLASSIFICATION

Another classifier is implemented to classify the objects, which is K-Nearest Neighbor. In this implementation, 3 nearest neighbors are used to classify the object, in which the 3 smallest distance from each class is summed up to get a more robust distance from each class and labeling is done based on the smallest distance from all of the classes [5]. KNN helps to reduce the effect of variation in illumination, shadow, and exposure change with translation and rotation and thus performed better than the nearest neighbor.

3.8. PERFORMANCE EVALUATION

To evaluate the performance of KNN classification, a confusion matrix is drawn showing the results of true labels versus classified labels. The matrix shows the classification results of 15 different objects.

KNN classification performed very well for objects having unique geometry and being able to classify them all the time. Those with similar 2D geometry sometimes get classified as the wrong class such as (Pen, Knife), (Circular Base, or Shoe Polish).

For testing, the object is kept at a random location with a different orientation from the training examples 5 times and the label is recorded to make a confusion matrix.

		TRUE LABEL														
		Heart	One	Knife	Two	Capital_H	Seven	Triangle	CircularBase	ShoePoslish	CardHolder	Diary	Calculator	Mouse	Trimmer	Pen
CLASSIFIED LABEL	Heart	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	One	0	4	0	0	0	1	0	0	0	0	0	0	0	0	0
	Knife	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0
	Two	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0
	Capital_H	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0
	Seven	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
	Triangle	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0
	CircularBase	0	0	0	0	0	0	0	4	1	0	0	0	0	0	0
	ShoePolish	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0
	CardHolder	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
	Diary	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0
	Calculator	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0
	Mouse	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0
	Trimmer	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
	Pen	0	0	1	0	0	0	0	0	0	0	0	0	0	0	4

Table 1. Confusion matrix showing classification result of true label versus classified label

3.9. LIVE DEMO

Video is recorded showing the object recognition in real-time. Below is the link:

https://drive.google.com/file/d/1NzIJf0nkcJdZGy-m4pWg6sAEw5MnZ1xm/view?usp=sharing

4. EXTENSION

4.1. MORE OBJECT CLASSIFICATION

As an extension, 15 different objects can be classified. The performance of the same can be seen through confusion matrix 3.8. The classification in 15 classes has performed well.

4.2. UNKNOWN OBJECT CLASSIFICATION

To provide flexibility to this implementation to add a greater number of objects in the database, an extension is added to the code to detect unknown objects based on a distance greater than a threshold. When an unknown object is placed in the workspace, based on the distance threshold, it is labeled as unknown, and a prompt is displayed in the terminal to press ‘n’ and add the label of an unknown object. Once the label is entered, a new label along with its feature vector is appended to the current database. Now if the same object is shown, then correct labeling is done.

A video is also recorded to show the real-time implementation. Below is the link:
https://drive.google.com/file/d/1tZ_3RBWp0QtrEwRYBDKZqQsUEMSpFkai/view?usp=sharing

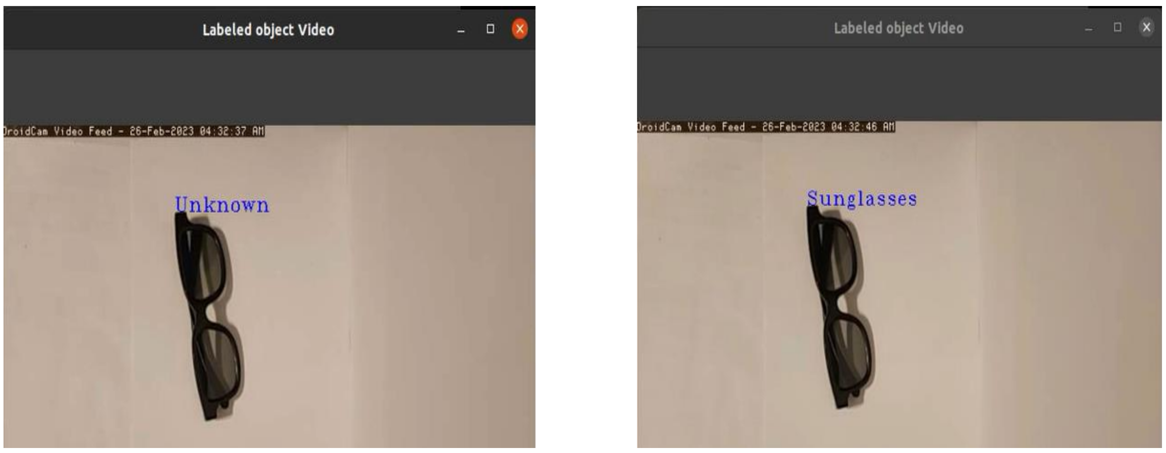

Labeled object Video
DroidCam Video Feed - 26-Feb-2023 04:32:37 AM
Unknown
Labeled object Video
DroidCam Video Feed - 26-Feb-2023 04:32:46 AM
Sunglasses

Fig. 6. Unknown object classification (left-before adding to database, right-after adding to database)

5. LEARNING

After completing this project, I learned about the concepts of 2-D object recognition and the various ways to achieve this. I have explored the area of morphological filters and ways to implement them from scratch as well as using OpenCV in-built functions. Also, I have learned about various classifiers to classify the image. The extension helped me to understand the need for flexibility in the code to add more objects as the system needs to learn automatically in a dynamic environment.

ACKNOWLEDGEMENT

1. https://docs.opencv.org/3.4/db/d8e/tutorial_threshold.html
2. https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html
3. https://docs.opencv.org/3.4/de/d01/samples_2cpp_2connected_components_8cpp-example.html
4. Zhihu Huang and Jinsong Leng, "Analysis of Hu's moment invariants on image scaling and rotation," *2010 2nd International Conference on Computer Engineering and Technology*, Chengdu, China, 2010, pp.V7-476-V7-480
5. https://docs.opencv.org/3.4/d0/d72/tutorial_py_knn_index.html