Project 3: Real-time 2-D Object Recognition

CS 5330: Pattern Recognition and Computer Vision

Arjun Rajeev Warrier

Abstract

In project 3, the goal is to develop a program that identifies an object placed on a white background. For this implementation, a video stream has been utilized from where, the user decides which frame to consider as a test or train case with the press of a button.

Experiment

The functionalities that were programmed as part of this project has been described below. This is a brief overview with short explanations and images that demonstrate the effect of these codes.

1. Threshold the input video

A threshold function was coded to separate a dark object from the white background and store as a binary image. For prepossessing, the source frame from the video stream is blurred and grey scaled before thresholding. Any pixel with a value below 120 (out of 255) is considered dark and a part of foreground.

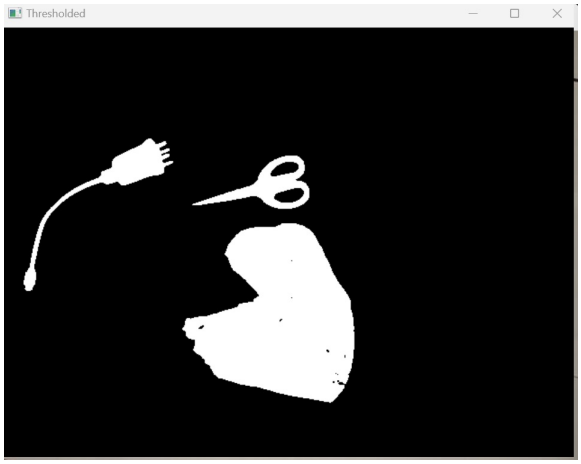

Figure 1: Thresholding. Background is black and foreground is white.

2. Clean up the binary image

8-connected dilation and 4-connected erosion are done in an alternating manner to get rid of spots in the foreground and holes in the background. This helps in cleaning up the image. OpenCV functions have been used here. A two-pass function has also been attempted.

Figure 2: Right side shows cleaned up binary image.

3. Segment the image into regions

With the help of the connectedComponentWithStats function, the binary image after being cleaned up is split into different segments with ids. A random color generator function has been written to assign different colors to different segments for easier interpretation. A two pass function was also attempted here.

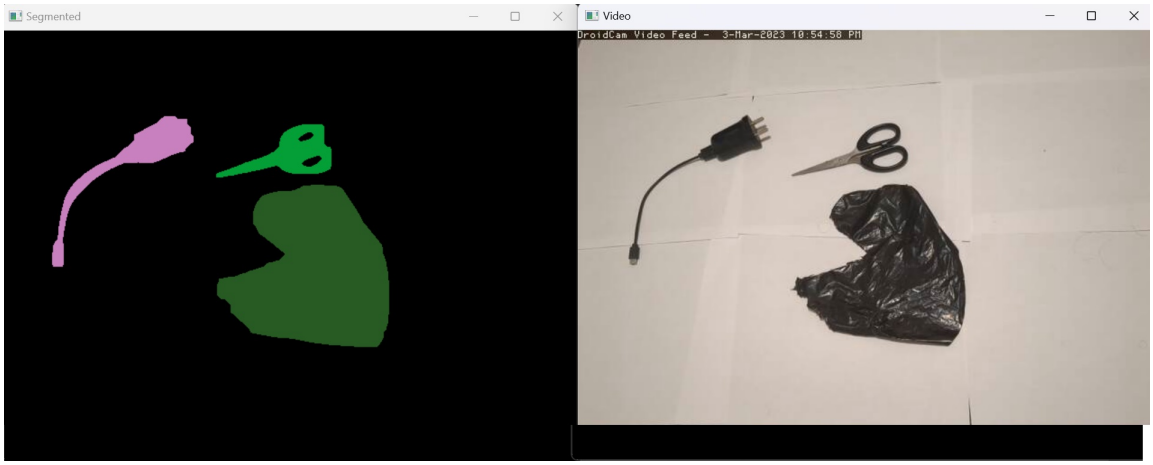

Figure 3: Different segments in different colours.

4. Compute features for each major region

Features used here are the moments computed from the moments function in OpenCV, the area of the bounding box used by the segment and the height to width ration of the box. These were then stored to a csv file that functions as the database for classification later.

The moments function used here, is to calculate the image moments of a given image from a 2D matrix. These moments are statistical measures that describe various properties of the image, such as its center of mass, spread, skewness, and orientation. These moments are useful for computing different properties of the image. For example, the area of the image can be computed using the zeroth order moment, the centroid can be computed using the first order moments, and the orientation of the image can be computed using the second order moments.

Figure 4: Bounding boxes.

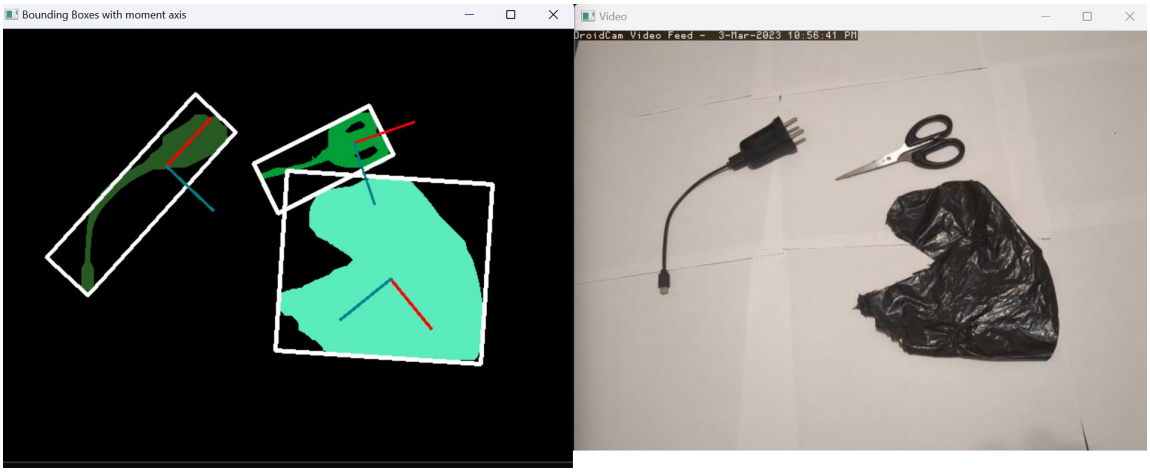

Figure 5: Bounding boxes with axes of moments.

5. Collect training data

The features computed and stored in the feature vectors are passed to a csv file using the csv util.h library from the previous project. For the purpose of this project, 10 objects were considered. When a user presses '6' on the number-pad, the program takes the current frame and segments it. A bounding box is drawn, moments are calculated and the segment to box ratio and the aspect ratio of the segment is stored as feature vectors against a user provided label in a csv file (database here).

6,7. Classify new images

With the nearest neighbour function, the program uses the previously trained database and com-

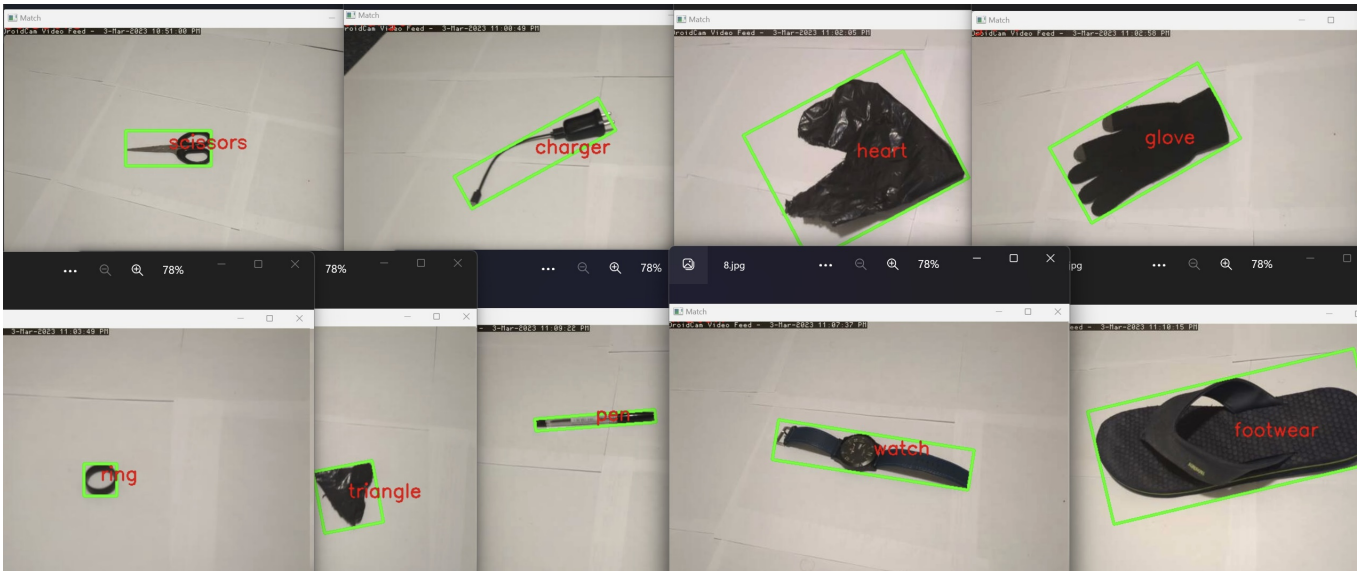

Figure 6: Different objects identified. Background is a couple of white papers.

putes the scaled Euclidean distances, [$(x_1 - x_2)/stdev(x)$], of the target image feature vector components from the corresponding stored feature vector components.
After which the distances are sorted, and the object in the database with the least distance from the targeted image is recognized as similar. This object name is then displayed in a separate window in the middle of the object bounding box.
A k nearest neighbour classifier has also been coded with k set a default value of 2. A slightly modified version of k-means has been used here.

8. Evaluating Performance

Confusion tables were recorded for the performance of both classifiers. Since, objects were mostly of distinguishable different shapes, the results were close to that of expected results. However, the glove and the cutouts of heart and triangle are somewhat similar in shape. Since, colour of the object is not recorded as a feature, these three have been confused with each other due to their similarity in shape.

K = 1		Classified Labels									
		Scissors	Watch	Ring	Footwear	Charger	Pen	Glove	Book	Triangle	Heart
True Labels	Scissors	2	0	0	0	0	0	0	0	0	0
	Watch	0	2	0	0	0	0	0	0	0	0
	Ring	0	0	2	0	0	0	0	0	0	0
	Footwear	0	0	0	2	0	0	0	0	0	0
	Charger	0	0	0	0	2	0	0	0	0	0
	Pen	0	0	0	0	0	2	0	0	0	0
	Glove	0	0	0	0	0	0	1	0	0	1
	Book	0	0	0	0	0	0	0	2	0	0
	Triangle	0	0	0	0	0	0	0	0	1	1
	Heart	0	0	0	0	0	0	0	0	1	1

K = 2		Classified Labels									
		Scissors	Watch	Ring	Footwear	Charger	Pen	Glove	Book	Triangle	Heart
True Labels	Scissors	2	0	0	0	0	0	0	0	0	0
	Watch	0	2	0	0	0	0	0	0	0	0
	Ring	0	0	2	0	0	0	0	0	0	0
	Footwear	0	0	0	2	0	0	0	0	0	0
	Charger	0	0	0	0	2	0	0	0	0	0
	Pen	0	0	0	0	0	2	0	0	0	0
	Glove	0	0	0	0	0	0	1	0	0	1
	Book	0	0	0	0	0	0	0	2	0	0
	Triangle	0	0	0	0	0	0	0	0	2	0
	Heart	0	0	0	0	0	0	0	0	1	1

Figure 7: Confusion matrices for two different classifiers.

Reflections

Some key takeaways for this project were:

- The effect of and need for pre-processing and clean-up of thresholded images before usage in classification were observed in this project. Without cleanup, the small reflections of the surfaces of objects were a constant hassle in segmentation and caused errors in feature vector computation. Blurring helped in reducing a large number of these disturbances as well as in getting rid of small objects that may have been seen as foreground by the program.

- Though unsuccessful, coding up the two pass algorithms have provided insight into how actual segmentation algorithms may work.

- This project has shown and helped in understanding the pipeline of object detection and how each step in the pipeline weighs in heavily.

References

[1] https://docs.opencv.org/

[2] Cunningham, Padraig Delany, Sarah. (2007). k-Nearest neighbour classifiers. Mult Classif Syst. 54. 10.1145/3459665.

[3] Cunningham, P. and Delany, S.J., 2021. k-Nearest neighbour classifiers-A Tutorial. ACM computing surveys (CSUR), 54(6), pp.1-25.