CS5330: Project 3 Real-time 2-D Object Recognition

Hussain Kanchwala **Abdulaziz Suria**

Summary:

In this project, we established a real-time 2D object recognition system. For ease of implementation, we have currently focused on dark objects on white background only. We have trained the system to recognize 13 different objects such as mouse, mug, power bank, watch, etc and provided the use the capability to add more objects to the recognition system. We have also enabled the user to click on a key and allow entry of new unknown objects into the database of our system and the system automatically asks the user to register the object if labelled as unknown. Above this we have created a dynamic confusion matrix to which the user can add more objects for real time system analysis. We have followed the process outlined in the assignment and they are listed out below:

1. **Custom:** OTSU Thresholding.
2. **Custom:** Custom cleaning/Morph filtering using Erosion/Dilation matrices
3. Image segmentation based on cleaned image.
4. Feature generation with 6 different features such as Hu moments, bounding box ratio, percentage fill, orientation of axis with least central moment **(visualized)**, etc.
5. Collection of data and preparation of training set.
6. Utilized different distance metrics such as scaled Euclidean distance and cosine similarity to classify new objects.
7. Confusion Matrix creation to evaluate performance of the system.
8. Also provided support using another classifier based on DNN and calculated the similarity score based on cosine similarity.

We have provided the required images and required insights further ahead in the report.

NOTE: We have used the real-time video option for object recognition

Original Images used for reference:

Figure 1:- Original Images

1. Threshold input video (CUSTOM)

We have created a custom OTSU threshold function which is a dynamic thresholding function which determines the threshold based on the maximum inter-class variance between background and foreground histograms after performing a 5x5 gaussian blur on the image. Here are the thresholded images for the original images.

Figure 2:- Thresholded Images

2. Clean up binary image (CUSTOM)

We have created custom dilation and erosion functions to perform morphological operations to clean the image. We have performed an 8x8 dilation followed by 12x12 erosion to clean the image.

Figure 3:- Cleaned Up Images

3. Segment Images into regions

We have utilized the connectedComponentswithStats() function of OpenCV library to get the segmented regions of the image. The user is asked to specify the minimum area of the region so that only the regions with area greater than it is further considered for processing the feature vector and are the only regions segmented.

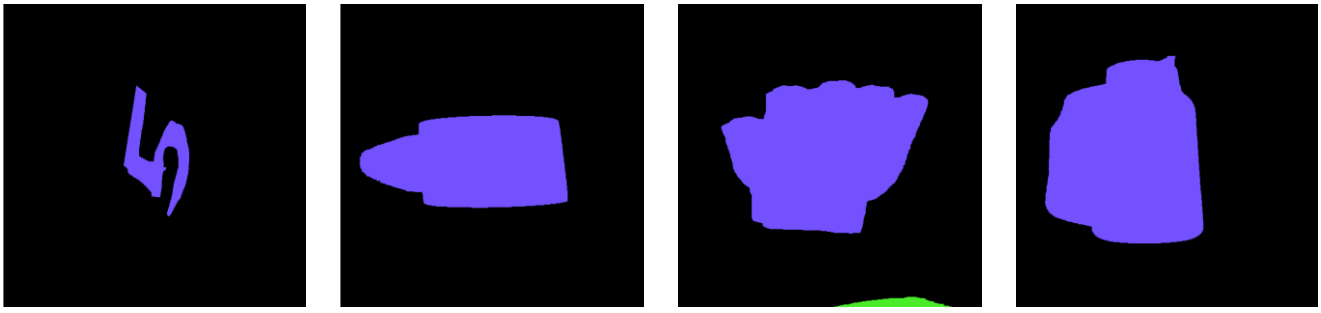


Figure 4:- Segmented Images as per Original Images Provided

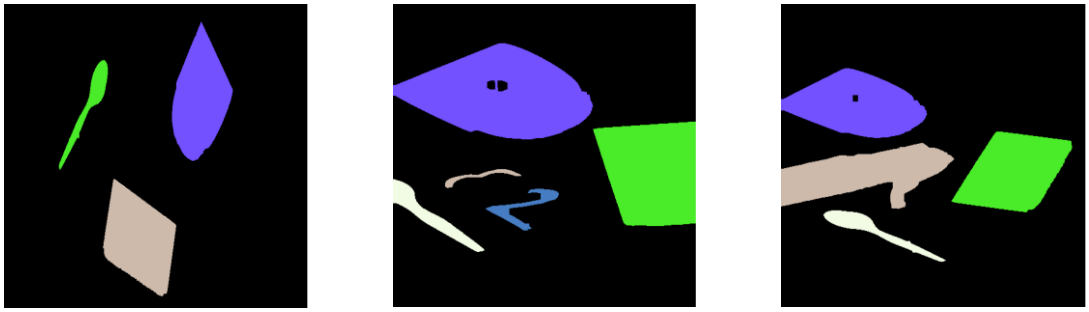


Figure 5:- Extra Segmented Images

4. Compute features for major region

Given the segmented region and region Id from the segmented image, the feature vector is generated for the major 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. 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 7 values, 5 are invariant moments, one of the rests is the percent filled and oriented boundary box ratio of height to width.

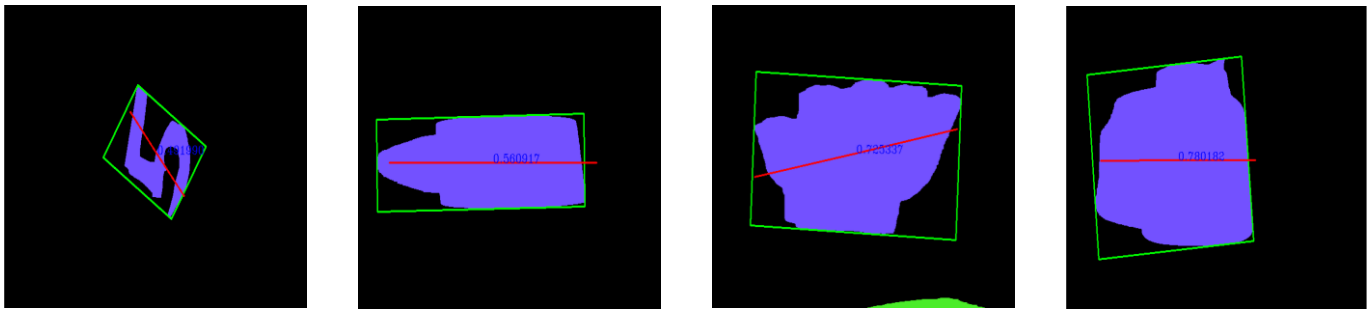


Figure 6:- Feature Calculation

mug,0.7389,4.6760,3.2015,5.0419,9.6384,0.7522,0.9734
five,0.4298,1.4105,2.9588,3.6141,7.2833,0.5226,0.6659
gloves,0.6884,1.8565,3.1762,4.1742,7.9003,0.7256,0.5680
cylinder,0.4882,1.1088,3.0042,3.3586,6.5402,0.6842,0.3509

5. Collect training data (As well as Unknown object extension)

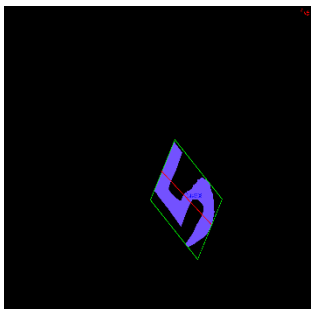
In our system, we use real-time video to perform the steps mentioned above and recognize the image if it is present in the database. If the error metric (scaled Euclidean distance) for the features of the image is above a certain threshold, it is classified as unknown, and the system asks the user to enter the label of the object to store it in the database. Other than that, the user can also press the 'N' key to register an object as well. If the image is present in the database and a similar image is shown, our system identifies the image correctly. A CSV file with a label followed by a vector of floats representing the feature vector of the image as the database of our system.

6. Classify new images

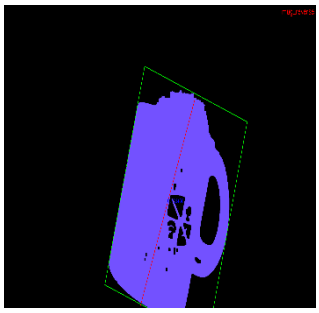
Euclidean distance metric [(x_1 - x_2) / stdev_x] is used to compare the features of the new object with the features of the objects stored in the database. If the minimum Euclidean distance

is less than 2 only then the object is classified as an object from database or else it is simply classified as unknown.

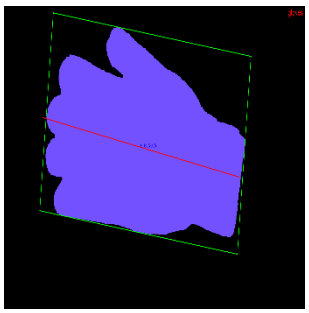
NOTE: - Watch on the top right for classification of the system.



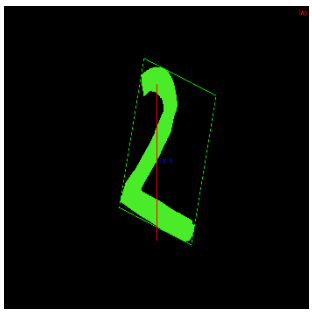
Pic. Five



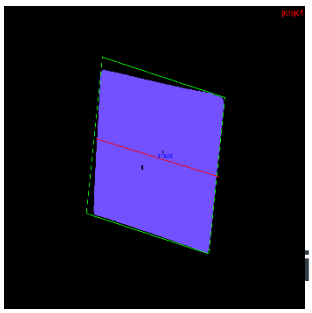
Pic. Mug



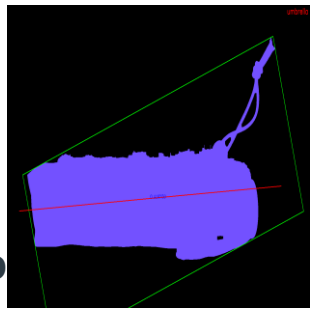
Pic. Glove



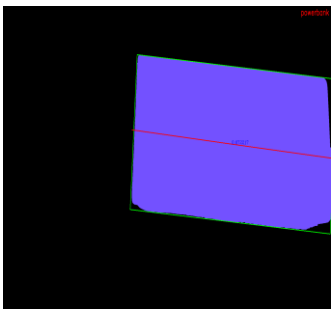
Pic. Two



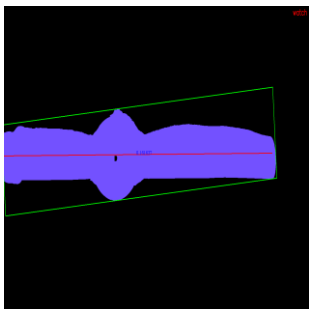
Pic. Passport



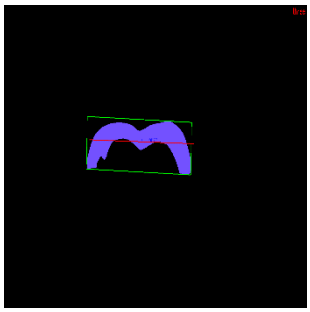
Pic. Umbrella



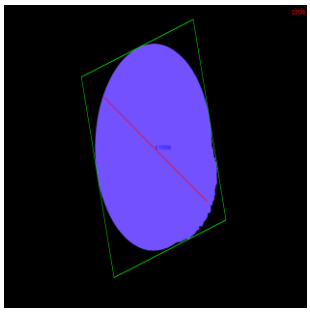
Pic. Power bank



Pic. Watch



Pic. Three



Pic. Circle

7. Confusion Matrix

We have created a function updateConfusionMatrix() which allows the user to assign a true label to the current object in the frame by pressing the ‘c’ key and takes the label given by the object recognition system as the predicted label adding it to the confusion matrix map with key as true label and values as predicted label and it’s count. We can press the ‘s’ key to display the matrix. This creates an extensible matrix and for the ease of purpose we are providing a confusion matrix on paper, but one can create a CMD based confusion matrix using the terminal.

True Label →

category	Spoon	watch	circle	five	glove
spoon	3	0	0	0	0
watch	0	3	1	0	0
circle	0	0	2	0	0
five	0	0	0	3	0
glove	0	0	0	0	3

5x5 Confusion Matrix

8. Demo video of the working system

Link to the video:
https://drive.google.com/file/d/1EJMEl4l5DKl0bxAUHKb7ezSuW1SneNh3/view?usp=drive_link

9. DNN classifier and comparison with Nearest Neighbour Method

If the classifier mode is selected as 2 then the DNN classifier starts. The user is required to provide the filename where the DNN embedding is stored or must be stored. The distance metric used to compare the two embeddings is cosine similarity. Rest the system to create new feature vectors is the same as above and the classification of objects as unknown is based on if the minimum error is greater than 0.3.

When it comes to performance the DNN classifier struggles at classification of between two and five, was continuously facing a problem in differentiating between a spoon and three as well as struggled to detect umbrella and cylinder. I also noticed that for the frame with two in it, if the two was rotated then the system immediately failed and started classifying it as unknown and the same followed for spoon as well. A 90-degree rotation of spoon from the point the embedding was recorded caused a miserable classification. I had to collect the embeddings of each type of object in different orientations to have the model classify it properly whereas in the case of nearest neighbour just one vector per object gave me a decent performance in case of classification.

I made a similar confusion matrix as the case for nearest neighbour to understand if the DNN was able to classify similar looking object properly.

Please watch the video below which adds to my explanation.

True Label →

category	Spoon	five	two	umbrella	cylinder	Three
spoon	3	0	0	0	0	3
five	0	4	3	0	0	0
two	0	1	2	0	0	0
umbrella	0	0	0	2	1	0
cylinder	0	0	0	3	4	0
Three	2	0	0	0	0	2

DNN CONFUSION MATRIX

Video Link (DNN):
https://drive.google.com/file/d/133kRO0Vl_BSWPvwWZX4B65rIqaoEw9nY/view?usp=drive_link

EXTENSION PT 1 (Detect Unknown Objects):

We have adopted a system which detects unknown objects, if the error metric is above a certain threshold the system starts classifying the object as unknown this can be seen in the videos. In addition to this if the object is classified as unknown then the system immediately prompts the user to enter the label for the unknown object to make a new feature vector and store it in the database. During the video trial I changed it to only with user presses 'N' start registration process by later again integrated the automatic system. Can be found in the code.

EXTENSION PT 2 (SUPPORT FOR more objects)

Both the DNN and Nearest Neighbour network are trained for 13 objects namely mouse, cylinder, umbrella, power bank, passport, watch, spoon, three, two, five, mug, glove, circle. This can be found in the .csv feature files attached in the submission.

Reflection:

Through this project, we gained a thorough understanding of various techniques and concepts essential for 2D object recognition. We understood different thresholding, cleaning techniques and how dynamic thresholding works. Also we learned the use of some inbuilt OpenCV functions which are used for segmentation and feature generation. We evaluated different features which we can use for object recognition such as moments, bounding box percent filled, orientation of least central moment, etc. We also learned how a basic 2D recognition system can work and compared it with a DNN based classification system. In our extensions, we also understood how unknown objects are added to the system and how we can support multiple objects as well.

Acknowledgements:

Thankful for the support from the TAs who helped in clearing our doubts during the Office Hours.

- https://docs.opencv.org/4.x/d3/dc0/group__imgproc__shape.html#ga107a78bf7cd25dec05fb4dfc5c9e765f
- OpenCV documentation