Homework 3

Real-time 2D Object Recognition

Cs 5330: Pattern Recognition & Computer Vision

Author: *Rishabh Singh & Hrithik Ketanbhai Kanoje*

Instructor: *Prof. Bruce Maxwell*
Department: Computer Science

Contents

1 Summary

This project is about real-time 2D object recognition. The goal is to have the computer identify a specified set of objects placed on a white surface in a translation, scale, and rotation invariant manner from a camera looking straight down. The computer should be able to recognize single objects placed in the image and identify the object an output image. If provided a video sequence, it should be able to do this in real time.
This project classifies the object based on their features. Before classifying it needs prepossessing such as thresholding, morphological filtering, segmentation of images into regions and computing features. The system has 2 modes. Inference mode and training mode. In Inference mode, the system can detect at a time 3 objects and show the label of each object. In training mode, the system allows users to label each item and store that class in the database.

1.1 TASK 1 : Threshold the input image

Inorder to perform thresholding first images needs to be converted into grayscale after that assessing each pixel and setting its value to either 255 or 0 if the pixel value is greater than or less than threshold value respectively. Images below are an example of thresholding. Refer to Fig.1

1.2 TASK 2 : Clean up the image

After performing thresholding images still has some noise and spots in the background to filter out that we requires cleaning of binary image by implementing morphological filtering using median filter to round of the larger structure and remove small structure in combine process. Morphological filters are used to clean up the image by filling the holes and remove the extra noise. We can remove this noise and spots by morphological filtering i.e. erosion using 4-connect neighbor and dilation using 8-connected neighbor. We have created opening function created using to perform erosion followed by dilation and closing function to perform three dilation followed by erosion. In our system, we have used closing three times by applying dilation three times and erosion one time. Refer to Fig.2

1.3 TASK 3 : Segment the image into regions

In this we need to segment the region. We have used connected component with stats function "**connectedComponentsWithStats()**" from OpenCV library then sort the areas to get N regions. Ignore too small regions and recognize the largest N and clean up frame to segment distinct regions in the frame. Each region after segmentation will implement the presence of separate object. Refer to Fig.3

1.4 TASK 4 : Segment the image into regions

In this task we need to calculate the features for a specified region in the frame. We calculate features of the largest region by using OpenCV's built-in function like moments() and HuMoments(). Image moments are a weighted average of image pixel intensities. moments(): computes a set of region attributes such as the centroid, size or area, moments, and central

Figure 1: Task 1: Original Image & Threshold Image

Figure 2: Task 2: Original Image & Clean Image

Figure 3: Task 3: Original Image & Segmented Image

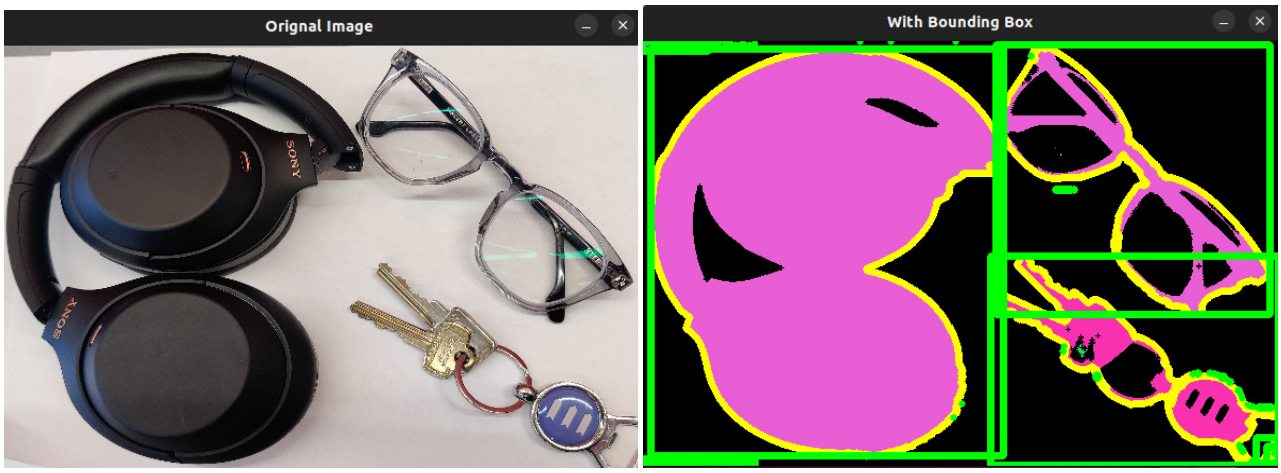

Figure 4: Task 4: Original Image & Bounding Box Image

moments. It is great that central moments are translation invariant. But that is not enough for shape matching. We would like to calculate moments that are invariant to translation, scale, and rotation.//
Fortunately, we can in fact calculate such moments and they are called Hu Moments.// Hu Moments (or rather Hu moment invariants) are a set of 7 numbers calculated using central moments that are invariant to image transformations. The first 6 moments have been proved to be invariant to translation, scale, and rotation, and reflection. While the 7th moment's sign changes for image reflection.
Furthermore, the system will only recognize the largest three objects, so the binder clip is ignored. Refer to Fig.4

1.5 TASK 5 : Collect training data

In this task, we must created a training dataset by capturing the features of an object, labeling them, and storing them in a csv file. Our training mode is activated by pressing the 't' key. If any new object comes then it creates a bounding box and when "t" is pressed then all the processing starts which gives us the segmented image and as the program quits its features gets

saved in the csv file, so that tye next time whenever the object comes again then its label will also come that will let us know that the object is in the database. The object in the frame is converted to a binary image, cleaned up with Morhpological filtering, and then segmented into regions. Following segmentation, the required region's features are calculated. When the key 't' is pressed, we can enter the label of the visible object in the frame, and the features and label are saved in a csv file.

1.6 TASK 6 : Classify new images

In this task we are using the Nearest Neighbor algorithm to classify new/unknown objects. The Nearest Neighbor algorithm finds the label from the training data whose features are the closest to the new object's features using Scaled Euclidean Distance as a Distance metric. According to the distance metric, the system returns the label of the most similar object. Refer to Fig.5

1.7 TASK 7 : Implement a different classifier

This task requires to implement a classifier other than scaled Euclidean distance. In this task, we used the K-Nearest Neighbors classifier with $K = 3$.
The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. In this classifier, we used Scaled Euclidean Distance as a distance metric. We calculate the distance of the feature of the object in the frame from the data in the training set using Scaled Euclidean Distance and sort them in ascending order, labeling with the smallest distance first, followed by the larger distances. When $K > 1$, the classifier chooses the label that is in the majority of the K Nearest Neighbors.

1.8 TASK 8 : Evaluate the performance of your system

1.8.1 Confusion Matrix for Baseline Classifier

Confusion Matrix	True Label					
	Objects	Pen	Glasses	headphones	Key	Watch
	Pen	10				
Predicted Label	Glasses		8	1		2
	Headphones		2	9		
	Key				10	
	Watch		2			10

1.9 TASK 9 : Capture a demo of your system working

The video recording linked below shows how our classifier detects different objects in real-time. Homework-3 Video

Figure 5: Task 6: Original Image & Classified Image

Figure 6: Extension 1: Original Image & Recognized Image

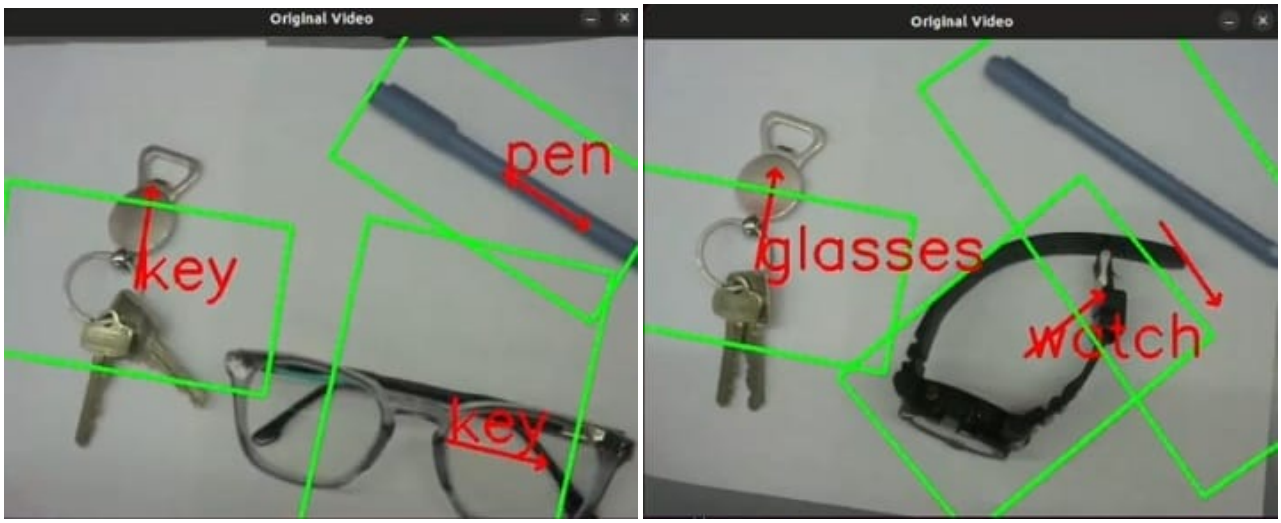

Figure 7: Extension 2: Identification of objects

1.10 TASK 10 : Extension

1.10.1 Extension 1 - Recognize multiple objects at the same time

After obtaining the regions of each object, the system will sort the regions by area, keeping the largest required number of regions. The system will then loop through the required number of objects, extracting the region as a separate image, calculating the features/taking training data, and displaying the class name in the video. Refer to Fig.6

1.10.2 Extension 2 - Determine whether or not the item is in the database

Set a threshold when classifying the object. When the Euclidean distance between the object and other objects in the database is large, the system knows that this object does not exist.
As a result, no class name is displayed, and the user must enter training mode to collect data about this object. Refer to Fig.7

2 Learning Outcomes

- Learned about working of Object recognition
- Learned about different types of classifiers
- Learned about various OpenCV commands such as moments, connected components and many more

References

[1] Connected components analysis. https://www.geeksforgeeks.org/python-opencv-connected-component-labeling-and-analysis/.

[2] Image processing - morphological filters - kdnuggets. https://towardsdatascience.com/image-processing-class-egbe443-6-morphological-

[3] Knn classifications. https://medium.com/hacettepeaiclub/k-nn-classification-in-c-1f8e6b16be25.

[4] Opencv: Basic thresholding operation. https://docs.opencv.org/3.4/db/d8e/tutorial_threshold.html.

[5] Shape matching using hu-moments. https://learnopencv.com/shape-matching-using-hu-moments-c-python/.