CS5330 – Project Report

Title: Real-time 2-D Object Recognition

Team: Aswin Chander Aravind Kumar, Shruti Pasumarti

TABLE OF CONTENTS:

1. OVERVIEW OF THE PROJECT:

The 2D object recognition project focuses on developing a real-time system for identifying specified objects placed on a white surface, regardless of translation, scale, or rotation, using a downward-facing camera. The project is divided into several tasks, beginning with thresholding to separate objects from the background, followed by morphological filtering to clean up the binary image and address any noise or issues introduced during thresholding. Connected components analysis is then employed to segment the image into regions, with region filtering implemented to focus on significant regions while ignoring small ones.

Feature extraction is conducted to compute translation, scale, and rotation invariant features for each major region, displayed in real-time on the video output. Additionally, the system is equipped with the capability to collect training data by storing feature vectors from known objects in an object database for later classification. Classification is achieved using a scaled Euclidean distance metric, enabling the system to classify new objects based on the closest matching feature vectors in the database and indicate the label of the object on the output video stream. Performance evaluation involves assessing the system's accuracy on various images of each object and constructing a confusion matrix to compare true labels versus classified labels.

Finally, a video demonstration showcases the system's functionality, while a second classification method, such as using a pre-trained deep network to create embedding vectors, is implemented to compare performance with the baseline system. Overall, the project combines image processing and machine learning techniques to develop a robust and efficient 2D object recognition system.

2. METHODOLOGY:

The methodology for Real Time 2D Object Detection adopts a comprehensive approach, integrating diverse techniques. Central to this methodology is the development of robust program capable of reading the target image from the database, extracting the features and finding the top matches which are similar to the target image. The detailed process involved in the implementation of the project is discussed in the following sections.

2.1 INPUT:

The live video stream from the camera can be used as the input for the program. The video device is opened and the frame from the camera is treated as the stream. The sample image of the video steam would look like the following:

Fig 1: Sample Image of the Video Stream

2.2 THRESHOLD:

Thresholding is an image processing technique that segments pixels based on their intensity values. It simplifies images by converting them into binary form, distinguishing foreground objects from the background. The project utilizes a thresholding algorithm to separate objects from the background in a video feed. Pre-processing steps such as blurring and adjusting pixel values enhance accuracy.

Dynamic thresholding using k-means analysis optimizes threshold selection. The system displays the thresholded video and records results.

Fig 2: Sample Video Image

Fig 3: Threshold Image

As we can see from the images the output video stream is a binary image with the background in white color and the foreground object in black color.

2.3 CLEANING UP BINARY IMAGE:

To refine the thresholded images, morphological filtering techniques are employed. By assessing noise, holes, or other imperfections, an appropriate strategy like erosion or dilation is chosen. Erosion shrinks object boundaries to eliminate noise, while dilation expands object boundaries to fill in holes, enhancing image quality for more accurate object recognition.

Fig 4: Sample Video Image

Fig 5: Cleaned Image

As we can see the images in both fig 3 and fig 5 there is a significant difference in the way the images are displayed as the latter is cleaner and better visualized.

2.4 SEGMENTATION:

Connected components analysis segments an image by grouping pixels into regions based on connectivity. It identifies distinct objects by labelling connected pixels with unique identifiers.

The system implements connected components analysis on the cleaned, thresholded image to identify regions. It ignores small regions and limits recognition to the largest N regions. Displaying these regions involves assigning distinct colors to each, optimizing visibility and clarity for simultaneous recognition of multiple objects. The segmented video is displayed real time and recorded.

Fig 6: T Sample Video Image

Fig 7: Segmentation Image

The larger regions are assigned a random color picked from the colour palette while the smaller regions are ignored. When the smaller regions are removed it will be easier to renumber the remaining region in the sequential order. The connected component analysis function of the OpenCV has been employed.

2.5 FEATURE COMPUTATION:

The system computes features for a specified region using the region map and ID. It calculates the axis of least central moment and oriented bounding box, overlaying them on objects. Features like percent filled and bounding box height/width ratio are computed, providing translation, scale, and rotation invariant analysis for object recognition.

Fig 8: Sample Video Image

Fig 9: Bounding Box around Image

2.6 TRAINING DATA:

The system implements a training mode triggered by a key press, enabling users to collect feature vectors from objects and assign labels. Upon pressing a designated key (e.g., 'N'), users are prompted for a name/label, and the feature vector for the current object, along with its label, is stored in a file for classification purposes.

```
aswinchanderaravindkumar@Aswins-MacBook-Air build % ./main live 51 0
Enter a label for the object: Specs
```

Fig 10: Label for saving the features

```
Region 1: Aspect Ratio = 1.85106, Percent Filled = 100%
Region ID: 1
Centroid: (674.342, 450.164)
Orientation: 51.9076 degrees
Region 2: Aspect Ratio = 0.764706, Percent Filled = 100%
Region ID: 2
Centroid: (584.093, 266.678)
Orientation: 88.3034 degrees
Region 3: Aspect Ratio = 0.84827, Percent Filled = 100%
Region ID: 3
Centroid: (1482.31, 440.415)
Orientation: 47.3058 degrees
Region 4: Aspect Ratio = 1.03254, Percent Filled = 100%
Region ID: 4
Centroid: (382.913, 612.209)
Orientation: 61.6013 degrees
Features saved to features.csv
```

Fig 11: Aspect Ratio, Percent Filled, Centroid, Orientation Calculations

Class	Region ID	Centroid X	Centroid Y	Orientation	Percent Fille	Aspect Ratio
specs	5	245.954	89.8391	0.154006	100	11.4
specs	6	126.675	74.7384	0.0967651	100	27.2424
specs	7	393.506	56.0162	0.756078	100	1.83333
specs	8	332.651	311.544	-0.148519	100	1.33403
specs	9	2.57937	3.62698	-1.0852	100	1.6
watch	1	595.079	235.444	-0.180163	100	0.75
watch	2	286.428	242.707	0.0450723	100	0.122654
watch	3	569.905	233.107	-0.185892	100	1
watch	4	544.844	231	4.68E-11	100	1
watch	5	520.43	228.979	-0.427588	100	0.8
watch	6	495.762	226.81	-0.502163	100	0.714286
watch	7	471.253	224.853	-0.655715	100	0.571429
watch	8	447.275	222.71	-0.554469	100	0.571429
watch	9	85.6479	194.504	-1.39427	100	0.12585
watch	10	316.613	244.558	-0.0280426	100	0.749609
watch	1	603.267	222.533	0.0848249	100	0.333333
watch	2	0	0	0	0	0
watch	3	0	0	0	0	0
watch	4	0	0	0	0	0
watch	5	579.5	219.867	-6.96E-11	100	0.666667
watch	6	556.128	217.051	0.111257	100	0.5
watch	7	533.5	214.5	1.5708	100	1
watch	8	510.5	211.5	1.5708	100	1
watch	9	487.606	209.061	-0.1947	100	0.666667
watch	10	465	206.5	-0.63967	100	0.6

Class	Region ID	Centroid X	Centroid Y	Orientation	Percent Fille	Aspect Ratio
wallet	1	0	0	0	0	0
wallet	2	297.379	192.501	-0.131371	100	0.749609
wallet	1	0	0	0	0	0
wallet	2	297.906	192.656	-0.127131	100	0.749609
wallet	1	0	0	0	0	0
wallet	2	0	0	0	0	0
wallet	3	0	0	0	0	0
wallet	4	298.543	193.419	-0.12372	100	0.749609
wallet	1	0	0	0	0	0
wallet	2	298.265	193.29	-0.124018	100	0.749609
j card	1	0	0	0	0	0
j card	2	305.5	317.583	0	100	0.2
j card	3	293	317.556	0	100	0.2
j card	4	0	0	0	0	0
j card	5	0	0	0	0	0
j card	6	272.863	312.972	1.34124	100	1.7
j card	7	303.878	298.366	-0.0146177	100	0.227273
j card	8	303.412	287.255	0.361825	100	0.5
j card	9	314.338	274.594	0.544679	100	1.77872
j card	10	332.812	201.781	-1.18107	100	1.14286
j card	11	288.817	215.578	1.18334	100	1.55245

Fig 12: Feature.csv file

The training system comprises two key functions: computeAndDisplayFeatures() and saveFeaturesToCSV(). In computeAndDisplayFeatures(), features are extracted from segmented regions within input images. For each region, features such as centroid coordinates, oriented bounding box, aspect ratio, and percent filled area are computed and stored in a 2D vector. Additionally, visual representations of these features are displayed, aiding in the understanding and verification of the feature extraction process. Subsequently, saveFeaturesToCSV() facilitates the storage of extracted features into a CSV file. This function first checks if the file is empty and writes appropriate headers if necessary. Then, it iterates over the feature vectors, appending the class name followed by feature values separated by commas to each row. Non-finite values are handled by replacing them with "0". This process enables the generation of labelled training datasets suitable for object detection.

2.7 CLASSIFYING NEW IMAGE:

The system enables classification of new feature vectors using a known objects database and scaled Euclidean distance metric. Utilizing nearest-neighbour recognition, the system labels unknown objects based on the closest matching feature vector in the database, indicating the label on the output video stream for real-time object identification. The label is displayed on the video stream.

Fig 13: New Images classified using the system

2.8 EVALUATING THE PERFORMANCE:

The system's performance is evaluated. A 5x5 confusion matrix is generated, showcasing true labels versus classified labels. By analysing the matrix, the system's accuracy and effectiveness in object recognition across different scenarios are assessed and reported.

```
Confusion Matrix elements:
1 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

Fig 14: Confusion Matrix

The confusion matrix has the column and rows as the true labels and the classified labels. The confusion matrix can be used to assess the system effectiveness and accuracy.

2.10 DEMO OF THE SYSTEM WORKING:

The link for the demo of the system recording.

2.10 K NEAREST NEIGHBOR MATCHING:

The system implements a K-Nearest Neighbor (KNN) classifier with K > 1 as an alternative to nearest neighbor matching. Unlike the voting method, KNN matching considers multiple training examples for each object, enhancing classification accuracy. This approach broadens the scope of the classification algorithm, offering improved performance in object recognition tasks.

KNN is utilized to classify a feature vector by finding the K nearest neighbours from a set of known feature vectors and their corresponding class labels.

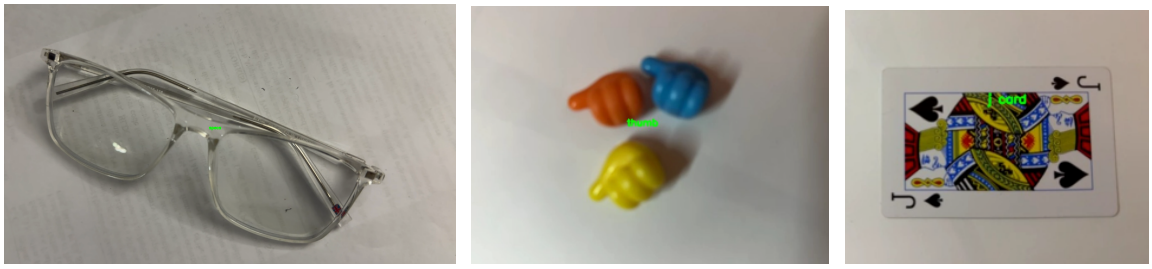

Fig 15: Classification using KNN

In this context, the K-nearest neighbor (KNN) algorithm operates by initially computing the Euclidean distance between the input feature vector (feature_vector) and each known feature vector in the set of known feature vectors (known_features). To ensure equal contribution from all features, the differences between corresponding feature values are divided by their standard deviations (std_devs). Subsequently, the distances, along with the indices of the known feature vectors, are stored in a vector of pairs (distances_and_indices), which is sorted in ascending order based on distances. The K nearest neighbors are then identified from this sorted list. By collecting the class labels corresponding to these neighbors, occurrences of each class are tallied using an unordered map (class_counts). Finally, the algorithm determines the majority class among the K nearest neighbors, selecting it as the predicted class for the input feature vector (closest_class). This method of classification is efficient and flexible, making KNN particularly suitable for scenarios where the decision boundary between classes is complex or unknown. However, the performance of KNN can be influenced by the choice of K and the representation of features in the dataset.

3. SELF REFLECTION:

Aswin Chander Aravind Kumar:

From my perspective, this project was an immersive journey into the realm of computer vision and object recognition. Beginning with setting up a workspace for real-time recognition, I delved into implementing essential tasks like thresholding, morphological filtering, and connected components analysis. Developing the system to compute features, collect training data, and classify objects added depth to my understanding. Challenges, such as fine-tuning parameters and optimizing algorithms, provided valuable learning opportunities. Finally, showcasing the system's functionality through a demonstration video was a rewarding culmination of the project. Overall, this hands-on experience enriched my skills in computer vision and problem-solving in real-world applications. But the implementation of classification process, training of the system, drawing bounding boxes proved to be very challenging and hard as I had to spend an extended amount of time on the project.

Sruthi P:

Despite having an early start on the task, I found myself concentrating on less impactful aspects of the implementation, which resulted in inefficiencies and extended the time required to complete the task. My primary challenge with this task was achieving modularity. I typically aim to write code that is easily understandable for others, but I recognize that this current code includes many repetitive elements that could have been streamlined by executing certain operations just once and reapplying them as needed. I intend to focus on improving this aspect of my code. I encountered significant difficulties with Task 5, particularly with correctly saving the computed features. Ultimately, I approached the problem by initially establishing different categories and then mapping the computed feature values to their respective categories, which resulted in a more organized CSV file. Moving forward, I aim to enhance the accuracy of my testing model and explore solutions for data capture under poor lighting conditions.

4. RESOURCES:

1] Professor’s Notes [Notes]

2] Professor’s KNN Codes

3] Professor’s CSV Codes