CS 5330 Project-3

Saideep Arikontham

Title: Real-time 2D Object Recognition

Project Overview

This project focuses on developing a real-time 2D object recognition system that would be capable of identifying different objects from training data. The resulting system would be able to read the frames from a video stream, perform a series of transformations to identify regions and successfully assign a label to the object detected from the video stream.

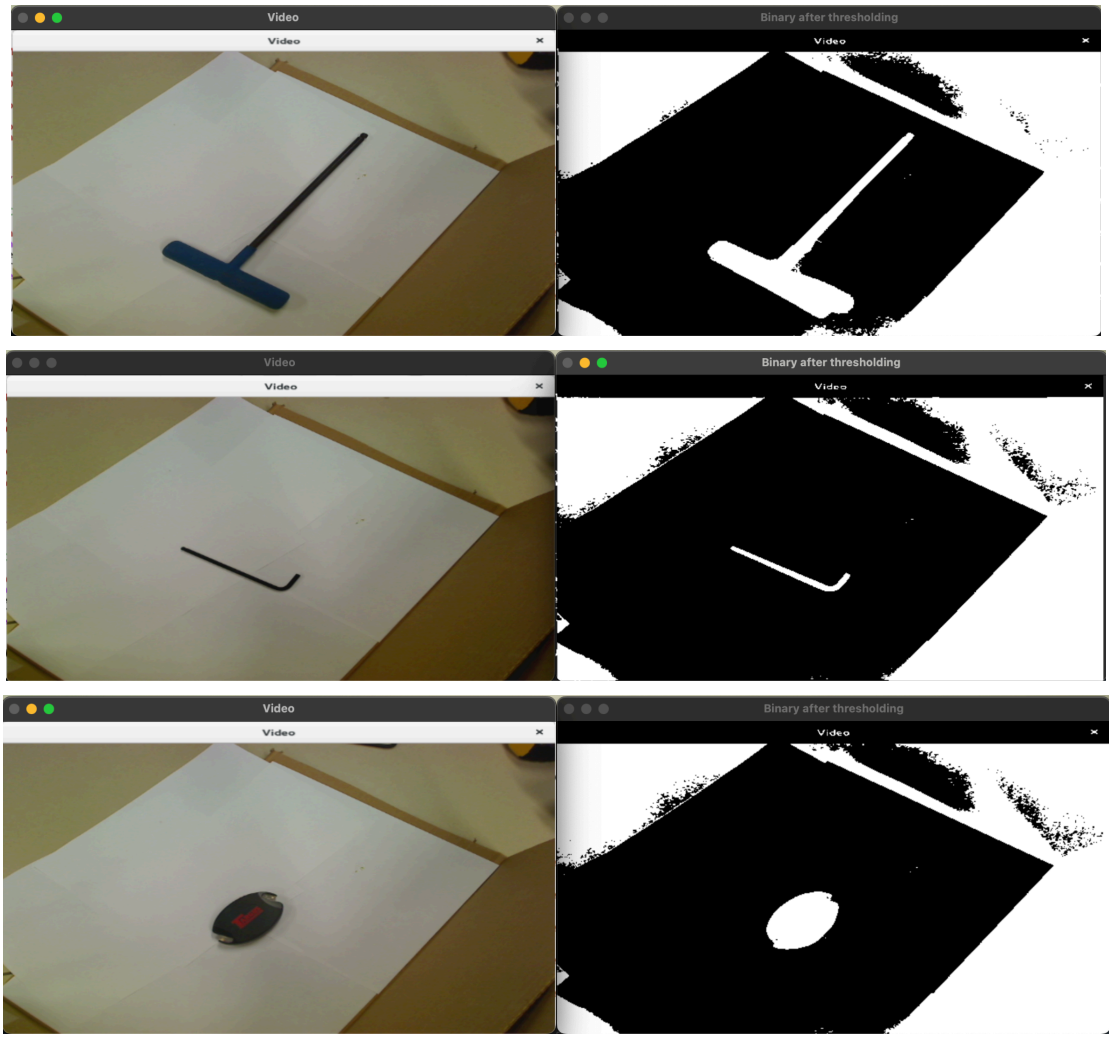
The exact process followed to build a real-time 2D object recognition system are listed below:

1. Make sure that you have your setup in a light background (white) with distinguishable dark objects (colors that can be easily identified).
2. After this, for each frame of the video stream, identify the background and foreground pixels. Convert your frame to a binary image where the background is black and foreground is white.
3. To reduce any noise, perform Morphological filtering to fill empty areas and remove small isolated pixels.
4. Divide the foreground regions from the frame into different segments and assign each of them a different color.
5. Now, for each frame in the video stream, identify the prominent segment/region and use a training dataset that has a label with different features like height-width ratio, bounding box filled percentage, etc.
6. To recognize the object, techniques like scaled euclidean distance, decision tree classification method have been employed.

Before extracting the feature set from any video stream, we will resize it to a fixed 640x360 pixel size to simultaneously understand the output of different stages. We first build the training data with respective feature information for each object to be recognized.

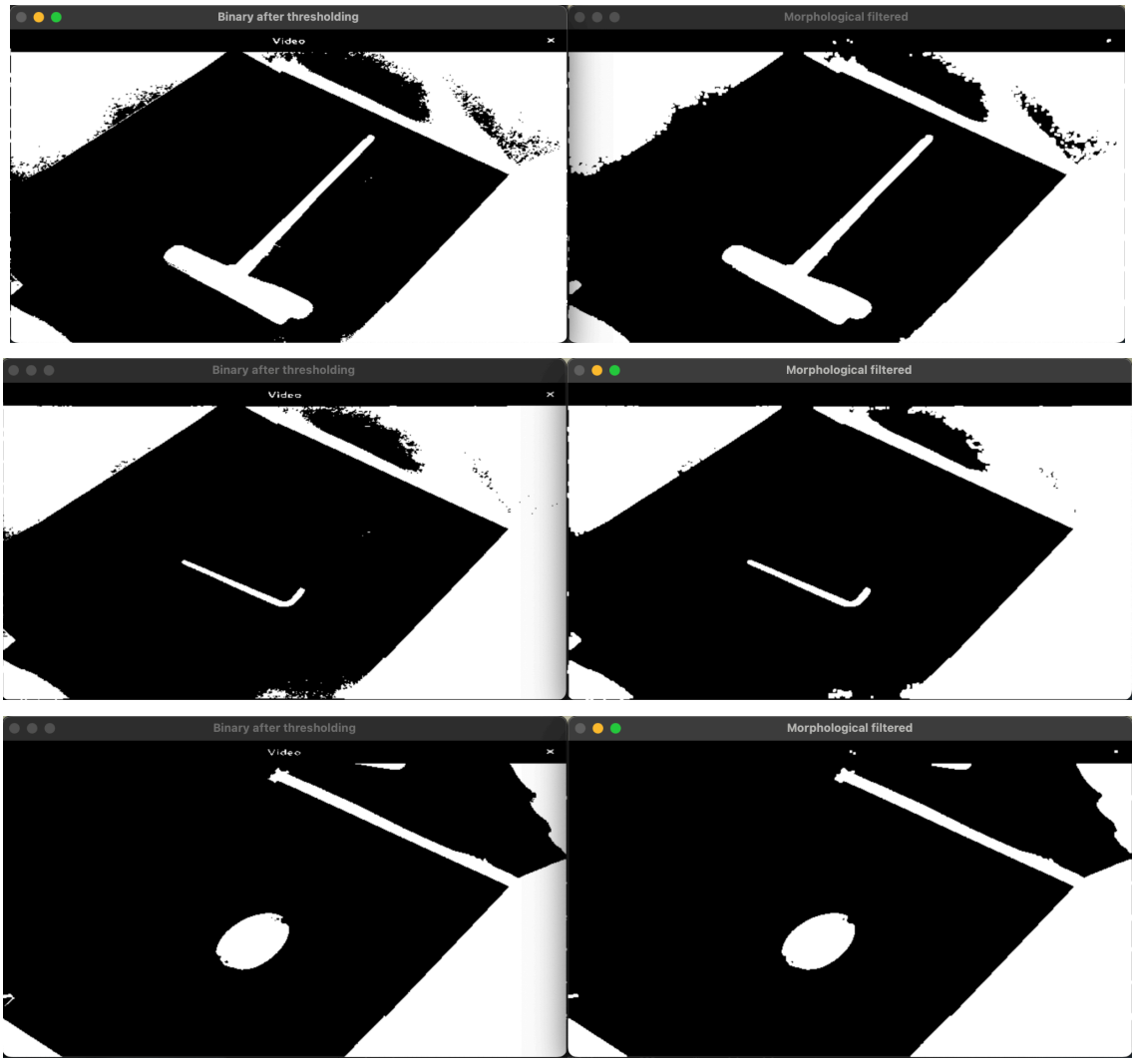
Directed Tasks and required images

1. **Threshold the input video (Req. result 1):** First, we take each frame from the video stream and apply thresholding. Without any preprocessing, I have applied a KMeans with K=2 and only considered 1/32 of the pixels in the image. This helped in successfully identifying means of the 2 dominant colors from the frame. Thresholding is applied to the frame by choosing a value in between the 2 dominant colors identified. Below are a few of the samples that are extracted from the video stream demonstrating the thresholding. For demonstration purposes, I am using individual images. *I have utilized the KMeans function given as part of class work and wrote my own code for thresholding and obtaining a binary result.*



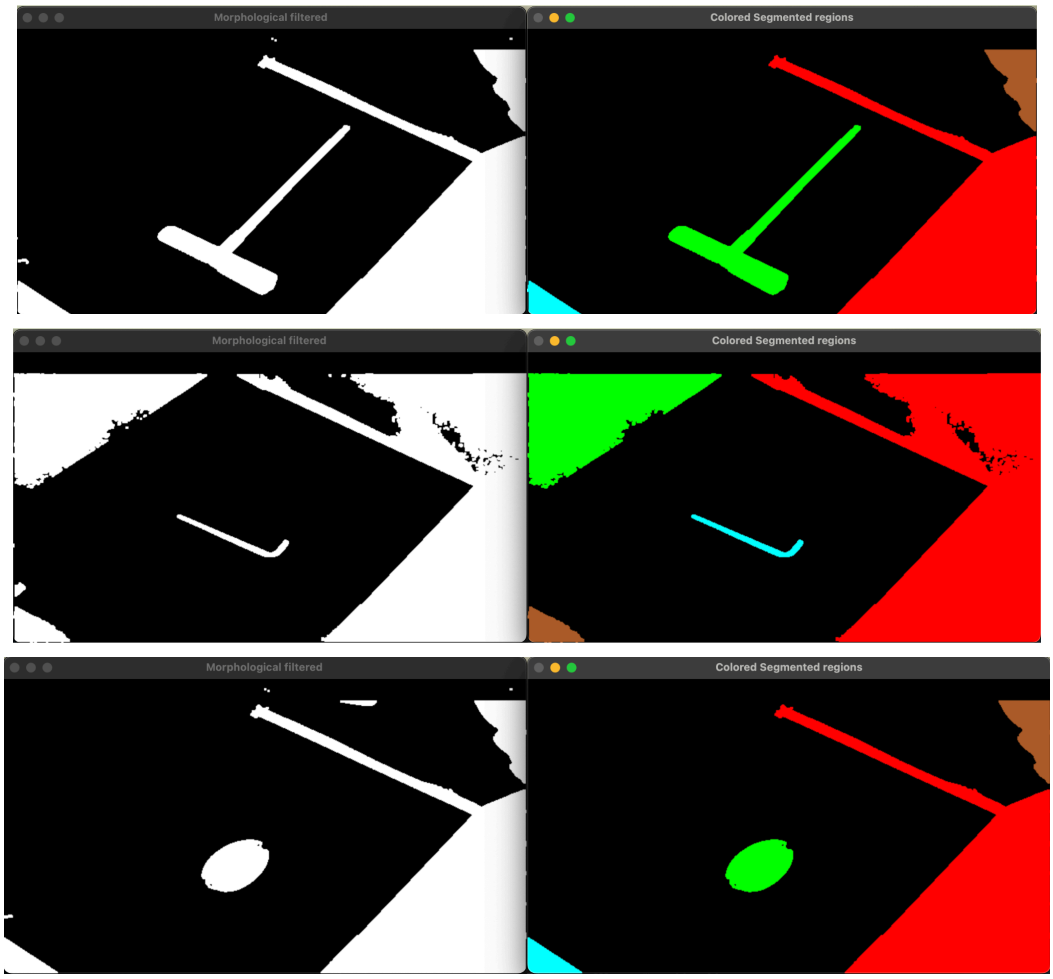
- We can see that the object in focus has been separated perfectly. But the region outside the white background is also thresholded as foreground, which we have to deal with later.
- We can clearly see the noisy pixels in the above output. The next step will help in removing those

2. **Morphological filtering (Req. result 2):** To eliminate or fill the small noisy pixels in the above thresholded outputs, we will use the technique called “Morphological filtering”. To achieve the desired result, I first performed Erosion (shrinking noisy pixels) using a 4-connected kernel method. To completely eliminate the small noisy pixels, I have applied 8 iterations of Erosion. Erosion is followed by Dilation (growing pixels) for 8 iterations with a 8-connected kernel. This helped in successfully eliminating the small noisy regions and filling a few empty regions in the objects as well. Below are sample outputs on still images. *I have implemented my own implementation of Morphological filtering (Erosion and Dilation). These functions are designed in such a way that they would work as expected irrespective of the kernel chosen for Erosion or Dilation.*



- We can compare the left (thresholded binary) and the right (Morphological filtering result) and see that the small noisy pixels have mostly been eliminated or grouped together.

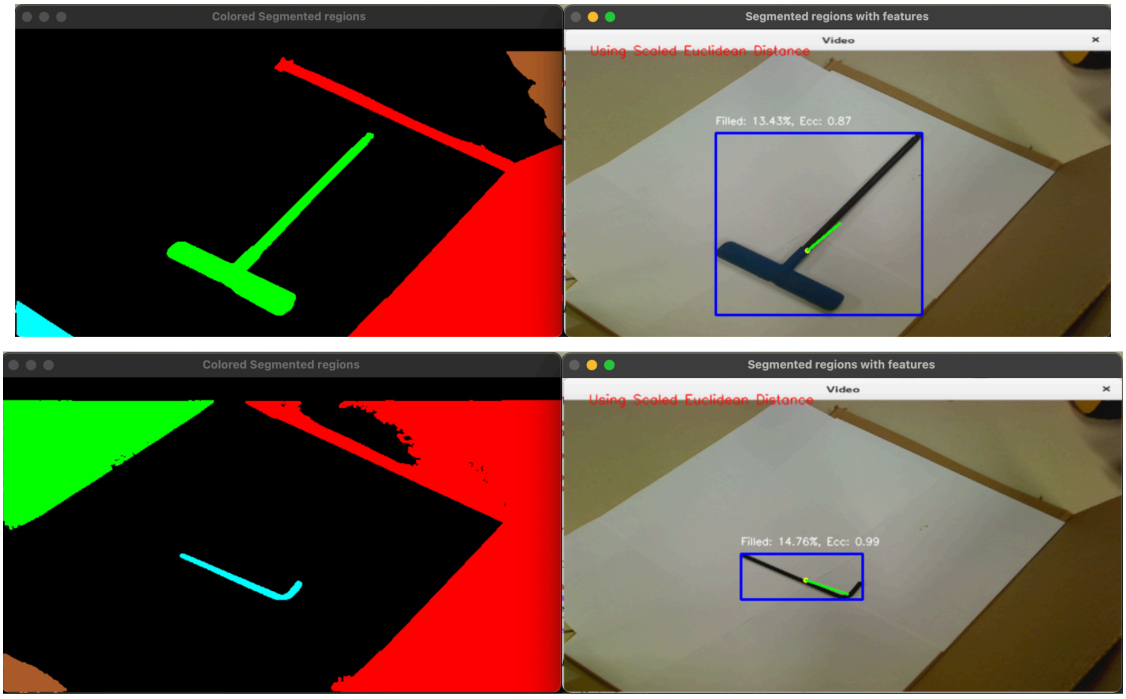
3. **Segment the image into regions (Req. result 3):** Now using the output of the above step, we can segment the image into different regions. OpenCV has a connecting components function and identifies regions, assigns a similar id to connecting pixels and returns these stats. Since there would be many regions within a given frame, I have limited the number of regions by ignoring any regions that have less than the defined threshold of pixels (I am ignoring regions with less than 500 pixels). Also, I defined an upper limit for the maximum number of regions that I want my system to identify (I am only identifying the top 5 largest regions apart from the background). All other regions that do not meet this criteria will be considered as background. I have also assigned a fixed color for nth largest regions. Red for 1st largest, Green for 2nd largest, Brown for 3rd largest, Cyan for 4th largest, Magenta for 5th largest and Black for the rest.

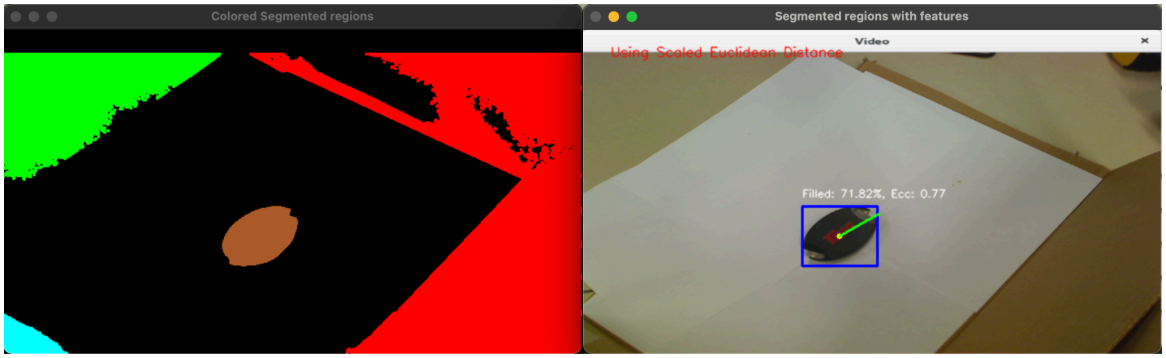


- We can see that the object is assigned a different color compared to the rest of the regions. Each region is clearly segmented and this output can be carried to the next phase.

4. **Compute features for each major region (Req. result 4):** Using the segmented output, we can compute features such as height-width ratio, boundary box percent filled, etc. below is the description of all the features that I computed for each frame.
    a. **Height-width ratio** the ratio of the bounding box height to its width. It describes the aspect ratio of the region, useful for distinguishing elongated vs. square-like objects.
    b. **Percent filled** is the proportion of nonzero (foreground) pixels within the bounding box. It indicates how much of the bounding box is occupied by the actual region
    c. **Orientation** is the angle of the axis of least central moment, indicating the primary direction of elongation. This helps determine the primary angle at which the object is oriented.
    d. **Eccentricity** is a measure of how elongated an object is, based on second-order image moments. It determines the elongation of the shape. Eccentricity is 0 for circles and close to 1 for elongated shapes.
    e. **Hu Moments** is a set of seven invariant moments used to describe the shape of an object. These features are scale, rotation and translation invariant and therefore help in better identifying the objects.

- I am only displaying the percent filled and eccentricity for the largest object in the output. Since there is a high chance that the largest object might be touching the border (window boundary). Therefore, if there is a boundary pixel in any region, I am ignoring that region. So, my system will only focus on the largest object that is completely inside the frame/window. Below are the sample outputs for more clarity.





- The title “Using Scaled Euclidean Distance” is to distinguish the method used to recognize the object.
- The yellow dot is the centroid, and the green line is the orientation axis. We can see the bounding box accompanied with the Filled percent and eccentricity.

5. **Collecting training data:** Now that I have all the necessary pieces, I proceed to collect training data. Now that I have the output as above, I have assigned a key “N” or “n” to be pressed to collect the features for the object in the current frame. Users will be prompted to assign a label for the object. The label along with its respective feature values are stored in a csv file which will act as our training data. As a part of my training data, I have selected 5 objects, for which I have collected 5 different images each. Below is the sample of how the csv file data looks like.

```
book,1.111111,0.585734,-1.129491,0.673027,-0.760129,-2.587129,-4.861118,-7.978805,9.999988,9.235203,-9.999988
```

- Label, height-width ratio, percent filled, orientation, eccentricity, hu moments (7) is the order of features stored in the csv file.
- Below are the images of the objects that I have chosen for training my system.
- **Perfume:**

- **Book:**

- **Mouse:**

- **Beanie:**

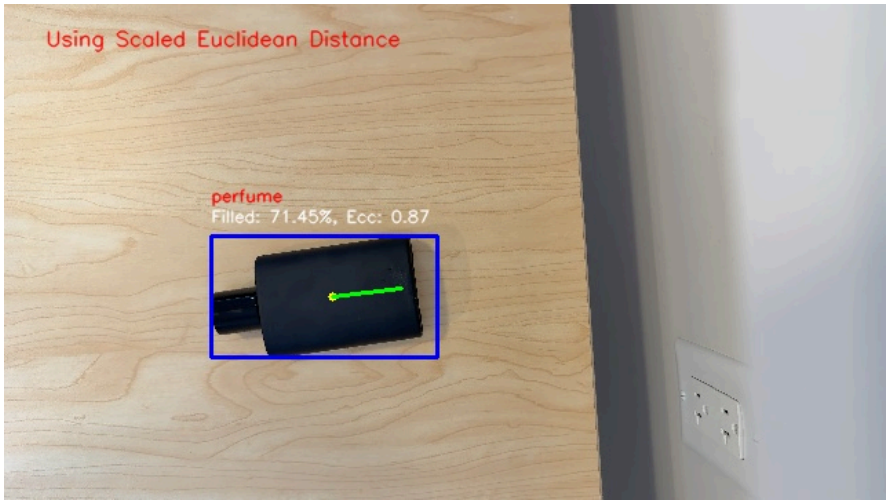
- **Box:**

- The images are chosen in such a way that they are clearly distinguishable from the background in terms of color and I also ensured that they have a unique shape when compared to each other.
- So, to reiterate, after placing the object in the frame and pressing “N” or “n”, the user will be asked to enter the respective label name for that object. Features will be calculated for that object and stored in a training data file along with the label entered.
- For my system, the above mentioned names are the label names.

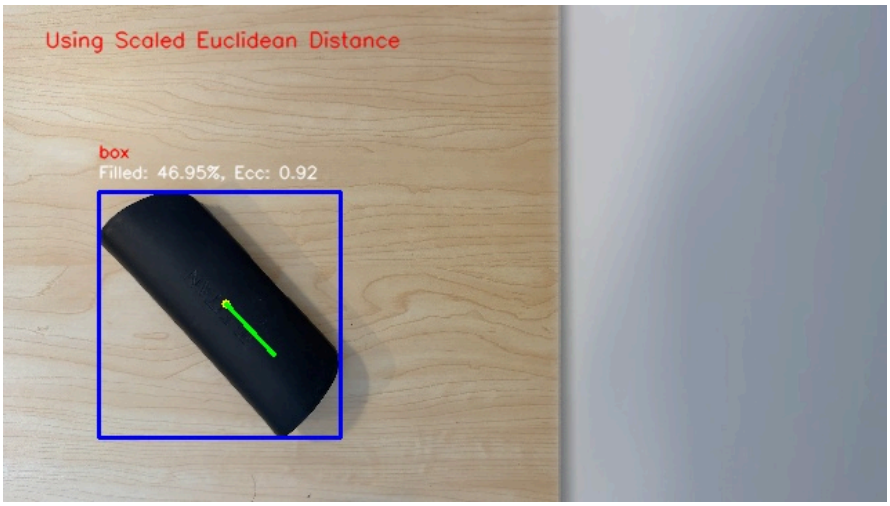
6. **Classify new images (Req. result 5):** Now that I have a training set, I can now recognize the objects using this. Now, the object we place in the frame will be our query object. This means that it is the object for which we want to identify the label. All the object data in the training data will be considered as the potential solution points. To identify the best label, we use “Scaled Euclidean Distance”.

- Scaled Euclidean distance is a variation of the standard Euclidean distance where each dimension is scaled to account for differences in units, importance, or variance across dimensions. The scaling factor used was standard deviation of the dimension.
- The lower bound result value for scaled euclidean distance is 0. This means that when the two objects are exactly the same (all the pixels are the same), then the distance between them is 0. There is no definitive upper bound result value for this metric.
- Therefore, from all the training data, the one that has the least scaled euclidean distance with respect to the query data is the best label for our query object.
- The system will only detect the largest object
- Below are the output images of the items being identified by the system.

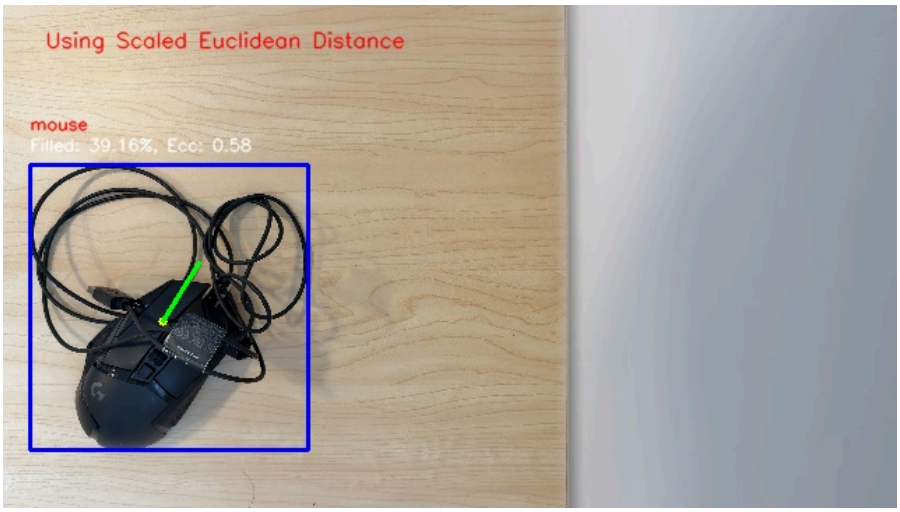
- **Perfume:**


Using Scaled Euclidean Distance
perfume
Filled: 71.45%, Ecc: 0.87

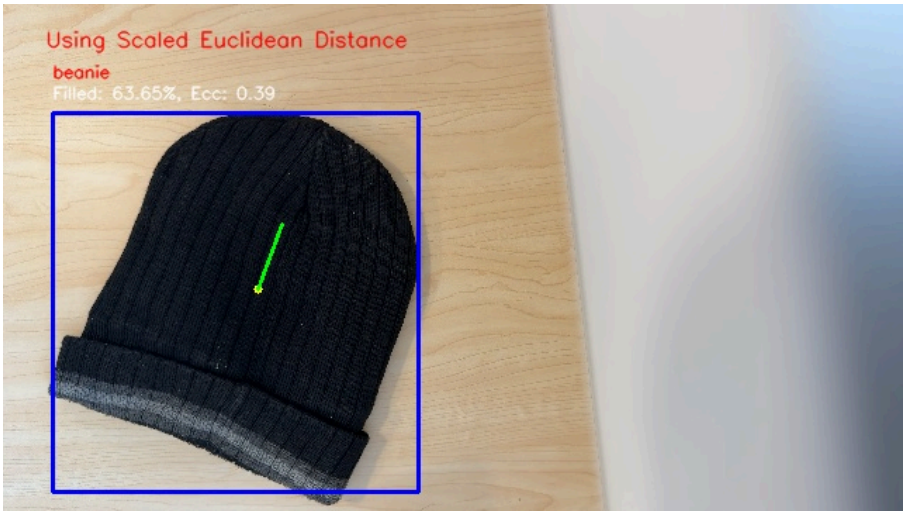
- **Box:**


Using Scaled Euclidean Distance
box
Filled: 46.95%, Ecc: 0.92

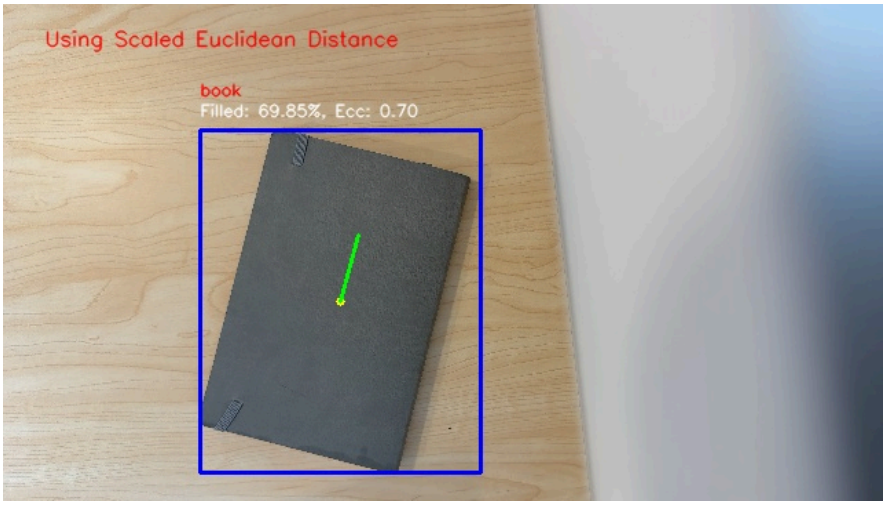
- **Mouse:**


Using Scaled Euclidean Distance
mouse
Filled: 39.16%, Ecc: 0.58

- **Beanie:**



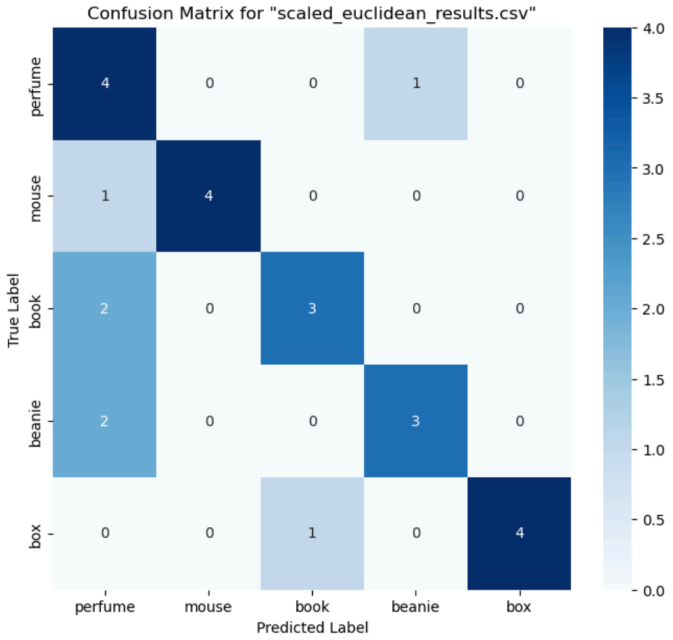
- **Book:**



7. **Confusion Matrix (Req. result 6):** I have tested the system on 5 different images for each object and stored the data into a file called “scaled_euclidean_results.csv”. The data that is stored is the image path (the current frame being tested and saved), the true label (Users will be prompted to enter a true label) and the recognized label (system detects this). Users can click “S” or “s” to save the object detection result. The system is designed in such a way to directly get the largest objects .Below is the sample of the csv file.

```
./result_images/img_name_1740080848.jpg,perfume,perfume
./result_images/img_name_1740080867.jpg,perfume,perfume
./result_images/img_name_1740080904.jpg,perfume,beanie
./result_images/img_name_1740081014.jpg,perfume,perfume
./result_images/img_name_1740081245.jpg,perfume,perfume
./result_images/img_name_1740081277.jpg,box,box
./result_images/img_name_1740081301.jpg,box,box
./result_images/img_name_1740081318.jpg,box,book
./result_images/img_name_1740081348.jpg,box,box
./result_images/img_name_1740081471.jpg,box,box
./result_images/img_name_1740081591.jpg,mouse,mouse
./result_images/img_name_1740081609.jpg,mouse,mouse
./result_images/img_name_1740081639.jpg,mouse,mouse
./result_images/img_name_1740081684.jpg,mouse,mouse
./result_images/img_name_1740081731.jpg,mouse,perfume
./result_images/img_name_1740081829.jpg,beanie,perfume
./result_images/img_name_1740081848.jpg,beanie,beanie
./result_images/img_name_1740081892.jpg,beanie,beanie
./result_images/img_name_1740081922.jpg,beanie,beanie
./result_images/img_name_1740081955.jpg,beanie,perfume
./result_images/img_name_1740081971.jpg,book,perfume
./result_images/img_name_1740082022.jpg,book,book
./result_images/img_name_1740082068.jpg,book,book
./result_images/img_name_1740082084.jpg,book,perfume
./result_images/img_name_1740082112.jpg,book,book
```

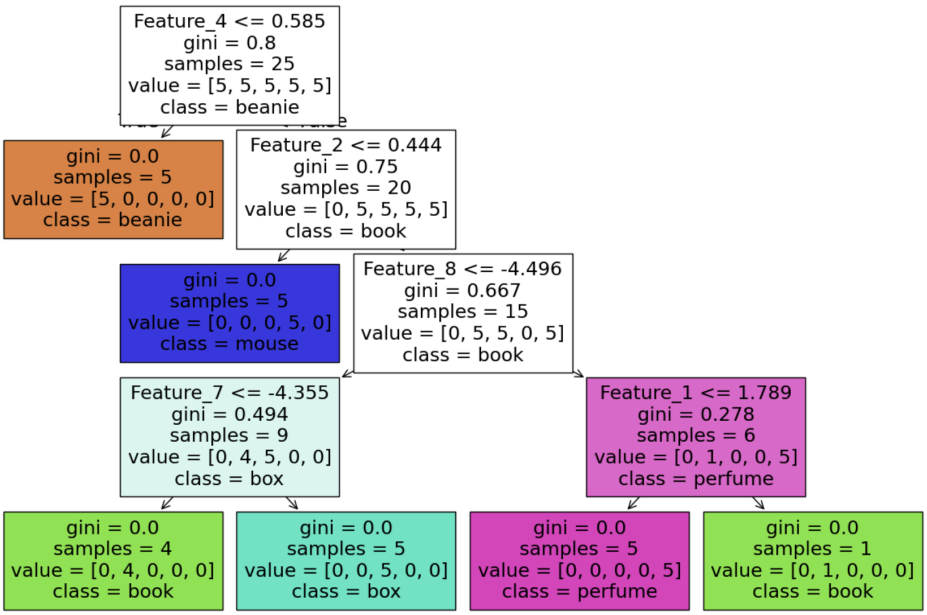
- I then used this csv file with python code separately to generate the confusion matrix using the true labels and the identified labels.
- Below is the confusion matrix for the above collected test predictions.



8. **Demo of working system (Req. result 7):** Below is the link to the video of live working of my system to classify objects using the scaled euclidean distance technique.
    - scaled_euclidean_distance_demo.mov (Anyone with a link should have access).
    - (https://drive.google.com/file/d/1TT2wmPORZ7fPrTAtXGV1Jyka17gutBhw/view?usp=drive_link)

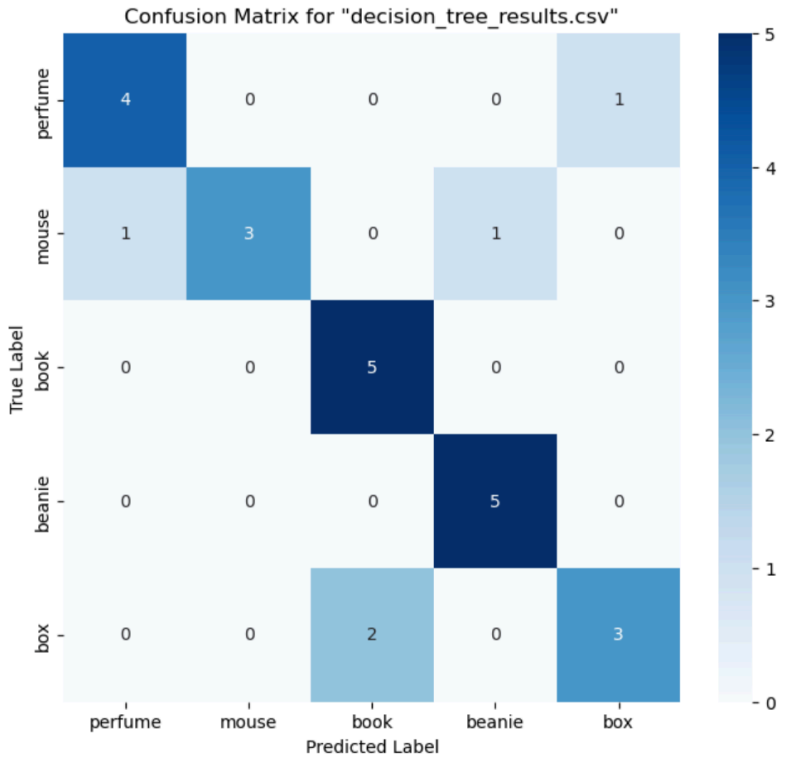
9. **Implement a second classification method (Req. result 8):** I have chosen building a Decision tree classifier as my second classification method. A decision tree works by recursively splitting the dataset into smaller subsets based on feature values, forming a tree-like structure. The goal is to split in such a way that similar class labels end up together.

- I have taken the training data csv file and trained a Decision tree classifier where label is the target and features are the input. The trained decision tree had a depth of 4. Below is the tree image that was a result of building a decision tree classifier.



- Using the decision nodes from the above tree, I wrote conditional statements in a function that my system utilizes to return the best label (predicted label).
- Since I have already implemented a working system for Nearest neighbor using scaled euclidean distance, I have integrated key press to implement the method of choice.
- By default, the system starts with the usage of “Nearest neighbor using scaled euclidean distance method”.
- For the system to use “Decision tree classifier”, the user must press “d” or “D”. To revert back to using the Scaled euclidean method, user must press “e” or “E”.

- I have designed the system to make sure that if the user wants to save the results (i.e., image path, true label, predicted label), these results will be saved to their respective files. “Scaled_euclidean_results.csv” for scaled euclidean method and “decision_tree_results.csv” for Decision tree classifier method.



- We can see that the errors in this case are not inclined to only one class like the scaled euclidean distance (Most of the predictions were perfume).
- Below is the overall accuracy of the two methods.
    - Scaled euclidean method test accuracy: **0.72**
    - Decision tree method test accuracy: **0.80**

- Therefore, we can conclude that the Decision tree method is more consistent compared to the scaled Euclidean distance method.
- Below is the link to the video containing a live video demo for decision tree classification.
    - decision_tree_demo.mov (Anyone with a link should have access)
    - https://drive.google.com/file/d/1iIBJdeHaQgDCEz2RrWwV_ZRVXuzlZr3P/view?usp=drive_link

---

Extensions

1. **Own Implementation of more than one stages of the system:** I have considered the suggestion to implement code for more than one of the stages in the system. I have written the code for both thresholding (threshold_using_KMeans function) and Morphological filtering (perform_morphological_filter function) without any opencv functions. I have also made sure that multiple methods can be integrated easily without any difficulty, with a simple key press.

- I use a mac and iphone so it was easier to use my mobile as a camera to implement a real-time object detection, directly detecting objects from a video stream.

2. **Detecting multiple objects at the same time:** I have built a system that will also be able to detect multiple objects in the frame at a time as well. A mode will be passed through the command line which will be either “single” or “multiple”. All the above methods are implemented in single mode. Using the “multiple” mode, I obtained the below result.

- Below is the link to a video demo of multiple object detection for both decision tree and scaled euclidean distance. (I did not have enough mount height for my mobile to fit both objects).
    - multi_object_detection.mov (Anyone with a link should have access).
    - https://drive.google.com/file/d/1ei2TMA44X1k_M43Ln030z4V0IUM6_fh0/view?usp=drive_link

- The system might not have worked as well as I intended it to for detecting multiple objects, but with more training data, I think the results would be more consistent and accurate as well.

Reflection

It was really great to work with real time object detection. I wish there was more time to explore a lot of other things but the time constraints made it hard to do so. I will definitely find time in the future to work on this in the future as I really had so much fun learning and implementing functions on my own from the understanding I gained through the classes. Co-ordinating realtime system with the resources that I had was also difficult as the video source could not be mounted at a high enough point for me to implement a more robust multi-object detection system. Apart from that, this was a great project to work on and dive deeper into how predictions can be made. This project showed the potential of using different features to obtain valuable information that will be vital in classifying the object.

Acknowledgements

The best resources I found for this assignment were ChatGPT and cloud recordings of classes. I used ChatGPT to understand errors that I encountered while working on this project. ChatGPT also helped me understand how few OpenCV functions work internally, different features that I can implement and other ideas that I could use to build a robust system. The cloud records helped me revisit the concepts of KMeans, thresholding, Morphological filtering, segmentation and more which made it easier for me to understand what the expected output would be for each stage of the project. The clear directions given in the project requirements made it easier to understand and implement all the tasks and obtain desirable results.