Report 3 – Real-time 2-D Object Recognition

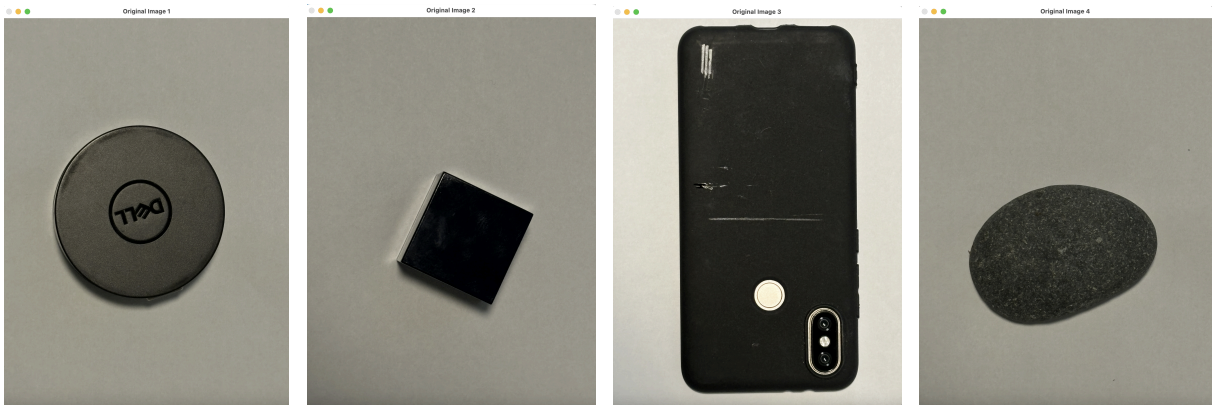
@Yanting Lai

Description

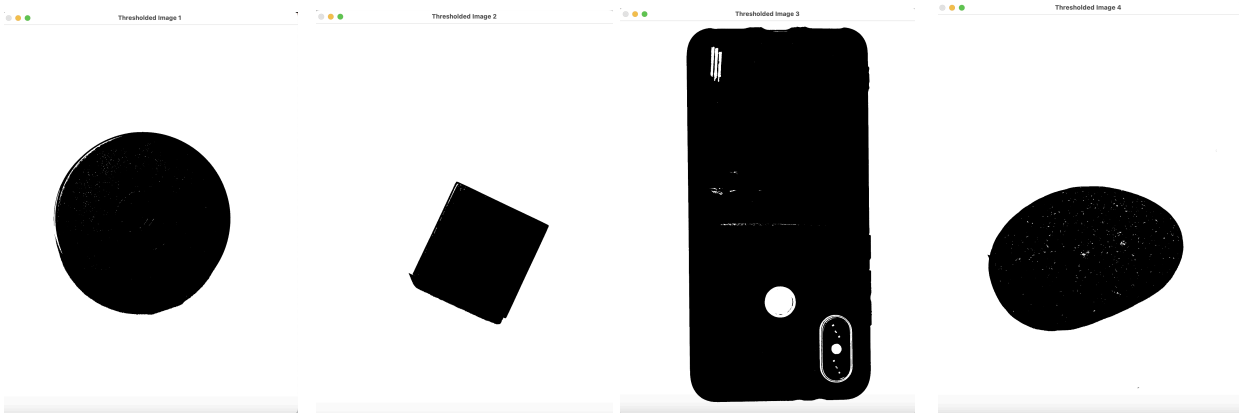
Task 1: Thresholding for Object Recognition

> This task involves implementing a thresholding algorithm to separate objects from the background. Objects should be darker than the background, with the area well-lit. The process includes pre-processing (blurring for uniformity), applying binary thresholding, and optionally using k-means to dynamically set the threshold. The result is a binary image that isolates the objects for further analysis.

Original image 1 Original image 2 Original image 3 Original image 4



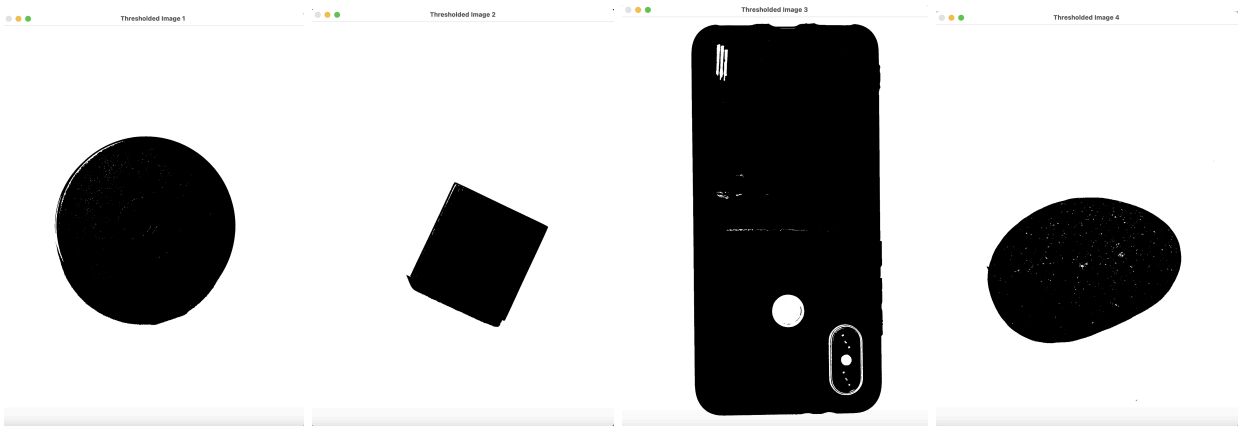
Threshold image 1 Threshold image 2 Threshold image 3 Threshold image 4



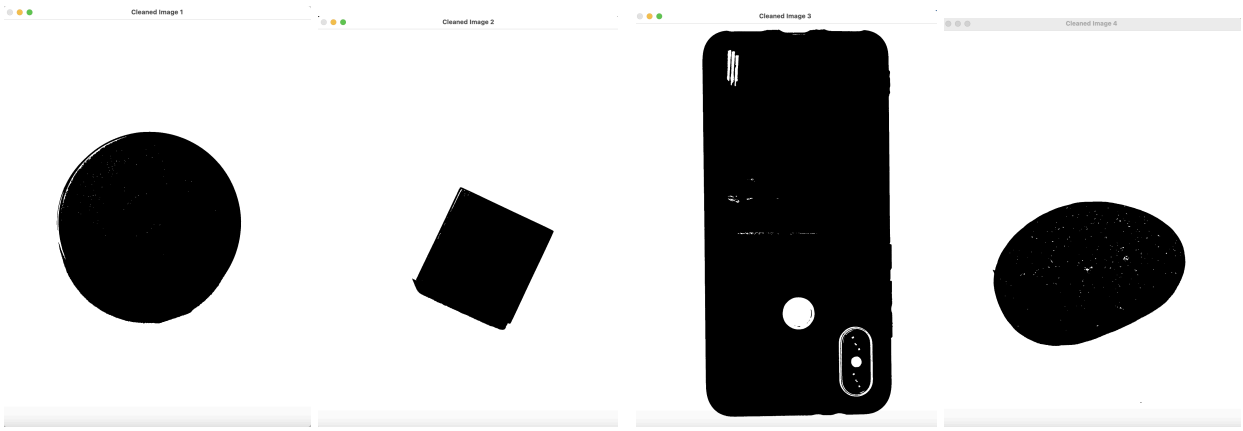
Task 2: Morphological Filtering

> In this task, morphological filtering is applied to clean up the thresholded image. This process removes noise, fills small holes, and improves the shape of the objects. The specific operation used is dilation followed by erosion. The result is a cleaner binary image, ready for further analysis, such as connected components.

Threshold image 1 Threshold image 2 Threshold image 3 Threshold image 4



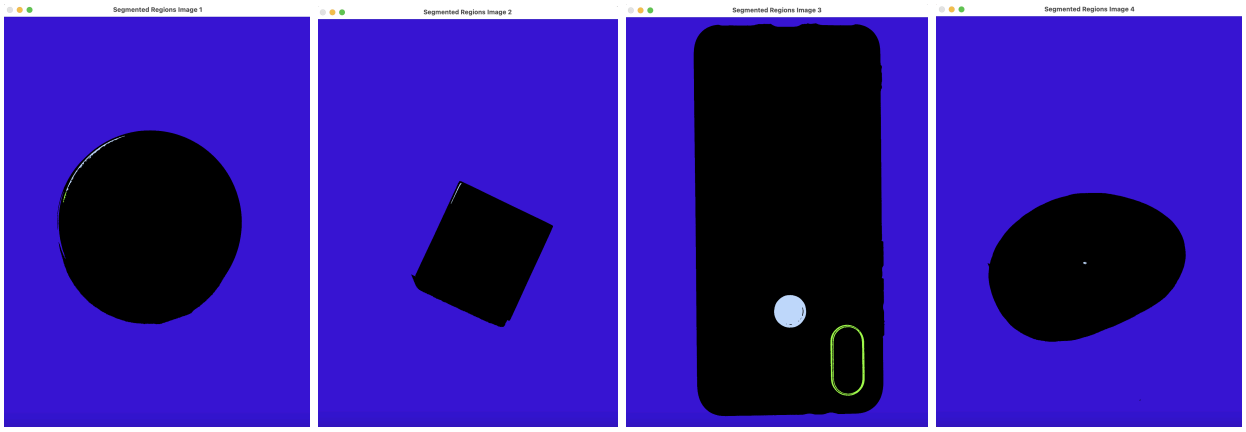
Cleaned image 1 Cleaned image 2 Cleaned image 3 Cleaned image 4



Task 3: Connected Components Analysis

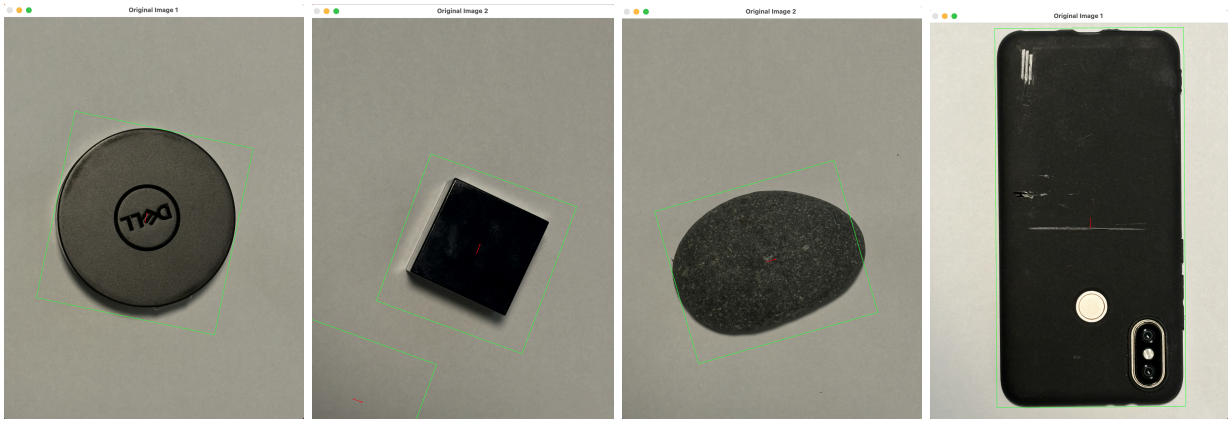
> Task 3 segments the binary image into distinct regions using connected components analysis. Each connected region is assigned a unique label, and small regions are filtered out. The result is a colored region map, making it easy to visualize and isolate individual objects in the image.

Segmented img1 Segmented img2 Segmented img3 Segmented img4



Task 4: Feature Extraction and Visualization

> In Task 4, we compute and display features for each connected component in the image. The features include the **centroid** (calculated using spatial moments), the **bounding box dimensions**, and the **percent filled** of the region. We overlay these features on the image by drawing the bounding box and centroid for each region, enabling visualization of object features in real time. These features are **translation, scale, and rotation invariant**, ensuring robustness for different object placements.



Feature_Vector_Summary

Image Filename	Hu Moment 1	Hu Moment 2	Hu Moment 3	Hu Moment 4	Hu Moment 5	Hu Moment 6	Hu Moment 7	Percent Filled	Height/Width Ratio
IMG_5572.JPG	0.159457	8.79511E-06	1.3505E-06	1.26857E-09	3.82743E-17	8.69222E-12	3.59456E-17	0.782161	1.0455
IMG_5573.JPG	0.166646	0.00026486	9.98167E-07	1.18088E-07	2.02194E-14	-3.17265E-10	3.51408E-14	0.540523	1.02021
IMG_5573.JPG	0.332921	0.0576936	0.0265846	0.00777364	0.000102801	0.00151287	4.38199E-05	0.25946	1.08724
IMG_5574.JPG	0.214894	0.0175902	1.15841E-06	2.50028E-06	4.24094E-12	3.30486E-07	3.47429E-13	0.939559	2.02471
IMG_5575.JPG	0.169287	0.0031692	4.60604E-05	1.50831E-06	5.28704E-12	2.69479E-08	-1.14061E-11	0.725292	1.17075

Task 5: Feature Extraction and Visualization

> Implement a training mode to collect feature vectors (e.g., Hu Moments, Percent Filled, Height/Width Ratio) from known objects, assign user inputted labels, triggered by a mouse click, and store the feature in the csv file.

To create the training data, I began by selecting **five distinct objects**: a nail clipper, a shaver, a candle lid, a remote control, and a mouse. For each object, I took 10 photos under a consistent lighting source, using a white background. I positioned and rotated each object differently in each photo to capture variations. This resulted in a total of **50 images for the training process**.

The system then loops through these 50 images, displaying each one to the user individually. As each image is shown, the user is required to **click on the image**. Upon doing so, a prompt appears in the terminal, asking the user to **input the label** identifying the object in the image.

After the user types in the label and presses "Enter," the system stores both the object's label and its associated features (such as Hu Moments, percent filled, and height-to-width ratio) in a file called **feature_vectors.csv**. The system then updates the displayed image to include the label superimposed on it, **visually confirming the labeling process**.

This process continues for each image in the dataset. After labeling one object, the system automatically proceeds to the next image, and the user repeats the labeling process until all 50 objects have been labeled and their feature vectors have been saved in the CSV file.

Some example of the training set's feature vector:

```
nail clipper,0.467087,0.19078,0.000853098,0.000568721,3.95574e-07,0.000242378,-2.11741e-08,0.373051,2.03095
nail clipper,0.468071,0.191503,7.85715e-05,4.6136e-05,2.76013e-09,1.90281e-05,-3.12285e-10,0.303184,1.45394
nail clipper,0.42221,0.150791,0.00148474,0.00098178,1.18535e-06,0.000381207,-2.86026e-09,0.761337,3.96214
nail clipper,0.406146,0.13787,0.00037979,0.000235609,7.04205e-08,8.64549e-05,2.87497e-09,0.504779,2.45028
nail clipper,0.452665,0.176832,0.000470051,0.000296897,1.10573e-07,0.000120723,-8.67497e-09,0.301771,0.801737
nail clipper,0.440728,0.166759,0.000460104,0.000291463,1.06561e-07,0.000115941,6.07005e-09,0.332901,1.55591
nail clipper,0.491985,0.209164,0.00570476,0.00421533,2.06705e-05,0.00192526,1.75465e-07,0.492302,2.89494
nail clipper,0.46019,0.18438,1.61412e-05,5.03066e-06,3.88595e-11,1.0891e-06,-2.33438e-11,0.955273,5.29715
nail clipper,0.428543,0.154498,0.00420358,0.00279449,9.57602e-06,0.00109589,1.8237e-07,0.300172,1.17537
nail clipper,0.463304,0.18582,0.00427482,0.00294048,1.04249e-05,0.00126679,9.03571e-08,0.314201,1.55634
mouse,0.189539,0.00933863,5.148e-06,1.86044e-07,-9.87459e-14,-7.26594e-09,-1.52968e-13,0.613114,0.998755
mouse,0.178425,0.0057166,2.26013e-05,2.05501e-06,1.13393e-11,1.23543e-07,8.21972e-12,0.839249,0.642347
mouse,0.190739,0.00957433,2.13877e-05,2.76546e-06,1.54644e-11,1.41936e-07,1.46012e-11,0.83376,1.67377
mouse,0.175519,0.00458621,7.22334e-06,1.06104e-07,9.10059e-14,7.04604e-09,-1.86091e-14,0.789846,0.703116
mouse,0.191482,0.0100964,9.51283e-06,5.35805e-07,1.20264e-12,5.3733e-08,1.30138e-13,0.666634,1.31965
mouse,0.191507,0.0099778,2.71701e-05,3.88808e-06,3.94402e-11,3.84566e-07,-6.43754e-12,0.873398,1.77287
mouse,0.181856,0.00684041,3.7008e-05,3.17017e-06,3.19773e-11,2.49388e-07,-1.25114e-11,0.65901,0.890336
mouse,0.191095,0.00986839,2.48228e-05,3.0789e-06,2.20654e-11,2.22424e-07,1.54149e-11,0.88402,1.79285
mouse,0.180992,0.00663689,1.89101e-06,2.48127e-07,1.24332e-13,1.65385e-08,-1.15887e-13,0.836895,0.643493
mouse,0.189077,0.0093886,1.69626e-05,1.80523e-06,6.69265e-12,1.04185e-07,-7.41611e-12,0.839206,1.68382
remote control,0.243041,0.0310958,4.01983e-05,1.54352e-05,3.82373e-10,2.72151e-06,4.01955e-11,0.837182,0.447137
remote control,0.251936,0.0360776,1.49136e-05,5.16859e-06,4.51932e-11,9.62586e-07,4.09504e-12,0.438758,1.12527
remote control,0.252047,0.0356464,4.44054e-05,1.80565e-05,5.11289e-10,3.36343e-06,-4.07675e-13,0.568314,0.623012
remote control,0.263323,0.0416977,2.76322e-05,7.51114e-06,1.07656e-10,1.52041e-06,-1.09305e-11,0.881758,2.53973
```

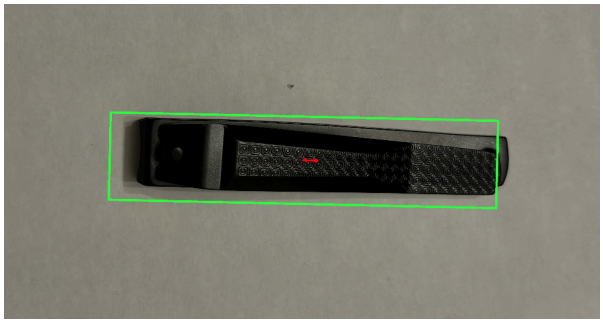
Task 6: Feature Extraction and Visualization

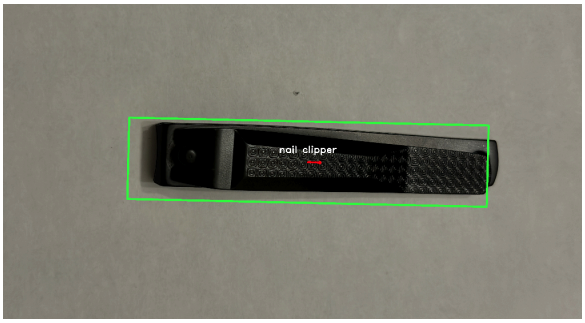
> For Task 6, the system classifies new images by comparing their feature vectors to the known objects in the database. It uses a scaled Euclidean distance metric, where feature differences are normalized by standard deviation. The object is labeled based on the nearest neighbor in the database. The assigned label is then displayed on the output image or video stream.

Distance calculation method: cosine distance

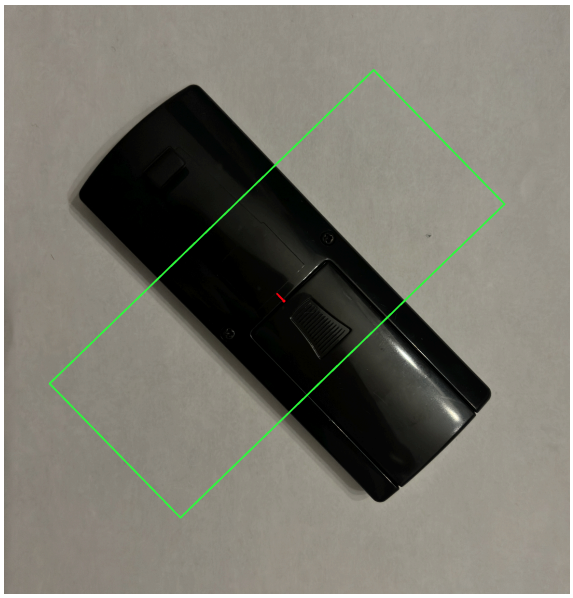
5 different objects, each object has three different photos from different rotations and positions, and using the cosine distance has successfully identified all the known objects correctly. In the below picture, I will show only time matching for each object and show the complete process in the video later.

Object 1 - Nail Clipper (correctly identified all the 3 times, here use 1 time as an example)

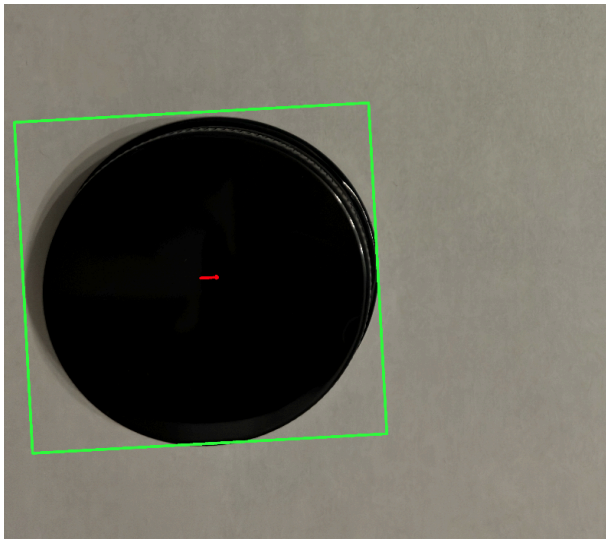




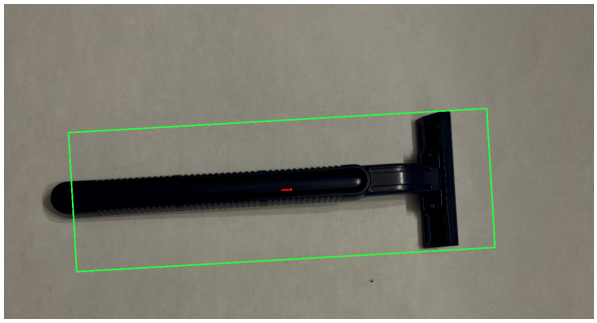
Object 2 - Remote Control (correctly identified all the 3 times, here use 1 time as an example)

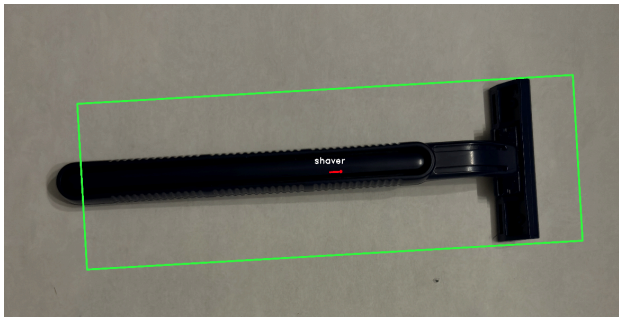


Object 3 - Candle lid (correctly identified all the 3 times, here use 1 time as an example)

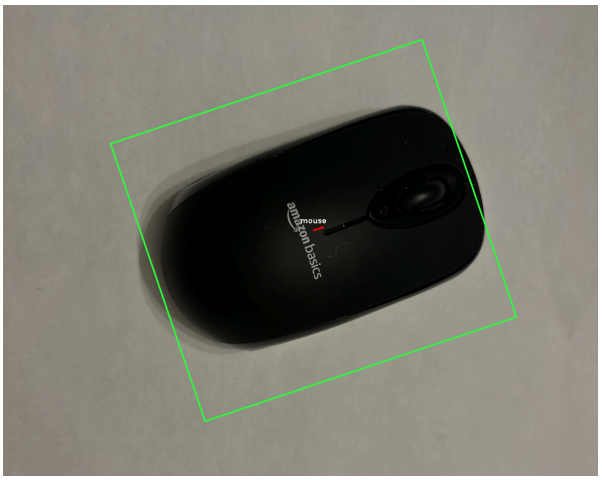


Object 4 - shaver (correctly identified all the 3 times, here use 1 time as an example)

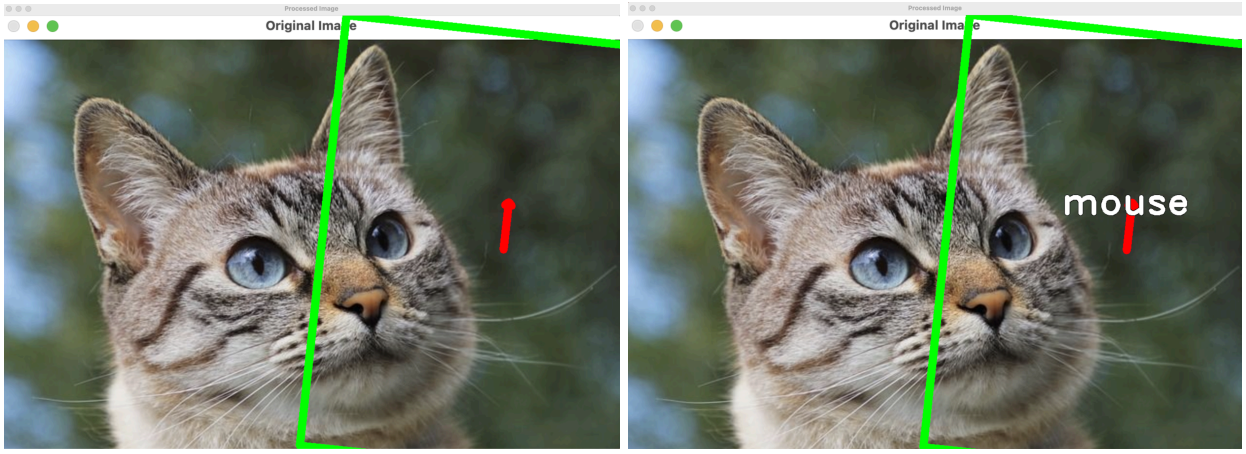




Object 5 - mouse (correctly identified all the 3 times, here use 1 time as an example)



Unknown object - currently being identified as mouse



Task 7: Feature Extraction and Visualization

> Task 7 implements an evaluation system to measure the accuracy of our object recognition program. It uses a confusion matrix to compare how well the system predicts objects versus their true labels. For testing, we used 3 different images of each object in various positions, and recorded whether each prediction was correct or incorrect. The 5x5 confusion matrix shows that our system achieved 93.75% accuracy, since we input an unknown object (cat) in the matching set and it was identified as a mouse.

Total Samples: 16 Correct Classifications: 15 Accuracy: **93.75%**

Error: the unknown object (cat) was misclassified as mouse

Confusion Matrix	Nail clipper	Remote control	Candle lid	shaver	mouse
Nail clipper	3	0	0	0	0
Remote control	0	3	0	0	0
Candle lid	0	0	3	0	0
Shaver	0	0	0	3	0
Mouse	0	0	0	0	3
Cat (Unknown)	0	0	0	0	1

Task 8: Video demo of system running and identifying objects

- Cosine distance matching
https://drive.google.com/file/d/1jEYPgzGMDvsY8pVEfPt3YOHCSHrS0n3U/view?usp=sharing

- KNN matching
https://drive.google.com/file/d/1PlgPZf-2Ty0EdX9CXmkpt3O5k8-nGGWi/view?usp=sharing

Task 9: A second classification method

> Task 9 implements another classification method. I implemented a K-Nearest Neighbors (KNN) classifier with K=5 as an alternative to the original cosine distance method. Testing showed that KNN achieved 68.75% accuracy, surprisingly lower than the original 93.75%, and the below paragraph will try to explore the reason why.

Total Samples: 16 Correct Classifications: 10 Accuracy: **62.5%**

Confusion Matrix	Nail clipper	Remote control	Candle lid	shaver	mouse
Nail clipper	3	0	0	0	0
Remote control	0	1	0	0	2
Candle lid	0	0	3	0	0
Shaver	0	0	0	3	0
Mouse	0	0	2	0	1
Cat (Unknown)	0	0	0	0	1

Why does the cosine distance method work better than KNN in this case?

The KNN's lower performance (68.75% vs 93.75% with cosine distance) can be attributed to several factors:

Feature Vector Composition: My feature vectors primarily consist of Hu Moments, which are designed to be rotation, scale, and translation invariant. Cosine distance naturally handles these invariant properties better.

Data Distribution: My dataset shows that classes like "mouse" and "remote control" have similar geometric properties (both are elongated objects). When using KNN with Euclidean-based distance, these similar shapes can cause confusion. Cosine distance better preserves the distinctive patterns in Hu Moments by focusing on the directional relationships rather than absolute differences.

Sample Size: With a relatively small training set, KNN's effectiveness is limited. Cosine similarity appears more robust with fewer samples as it captures the essential shape characteristics encoded in the Hu Moments more effectively.

Reflection

This assignment 3 again took more time than I thought. Here are several things that slowed me down:

1.Selection of the input images: At first I chose some pictures of cats and dogs that have flashy backgrounds without following the advice in the assignment. And that made the segmentation really difficult, let alone moving on to the object identification part. After learning from the professor, I learn that it's better to use: 1) single light source 2) on white background 3) object with black color 4) preferably don't have too much leverage

2.Handle the shadow of the image: I tried to develop a way to handle the image by trying out an adaptive thresholding method. Although the even easier way was to ensure there is no shadow in the input object(turn on the flash light). There are also other methods to consider: using HSV color space, and background subtraction.

3.Format of feature vector: I stored the label in the training set feature vector and it caused a format mismatch with the matching data set. I then realized I should use a hashmap to store the mapping relations between the feature vector and the label.

4.Overall I spent a lot of time on task 4 to figure out drawing the bounding box for the object. If I could redo the assignment again, I would choose to first understand the project from a high level perspective, understand what could be the tricky part in each task, ensure my assumptions about input are correct (wrong input, hard to get back later!!!!), ensure my implementations are efficient, and also be critical about the output. My model is simple, and my objects are simple, and my training data set is small, so in this case using a simple distance calculation can work better than using KNN method.

5.I am just glad I made it through.

Citations

[1] "Image Segmentation.". https://en.wikipedia.org/wiki/Image_segmentation

[2] "Image Moment." Wikipedia.https://en.wikipedia.org/wiki/Image_moment

[3] "OpenCV Documentation." .https://docs.opencv.org/

[4] "k-nearest neighbors algorithm."https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[5] "Cosine similarity.". https://en.wikipedia.org/wiki/Cosine_similarity

[6] "Morphological operations." https://en.wikipedia.org/wiki/Mathematical_morphology

[7] "Connected-component labeling." https://en.wikipedia.org/wiki/Connected-component_labeling

[8]"Image thresholding." https://en.wikipedia.org/wiki/Thresholding_(image_processing)