Project 3 Report

Name: Xujia Qin

Date: Feb 20th, 2025

Description:

The main goal of this project is to develop a real-time object recognition system that leverages the moment features of objects. The approach involves using a database of object features for recognition. Initially, I convert the colored images from the training set into binary images through thresholding to distinguish between the background and foreground. Image segmentation is then applied to identify the object's region, and the moment features of this region are calculated.

For classification, various distance matrices and classifiers are utilized to compare the attributes of the target object with those in the training set to help determine the most similar object, which is then assigned as the label for the target. For each object, 3 photos with different angle and orientations are utilized to assess the system's performance during testing.

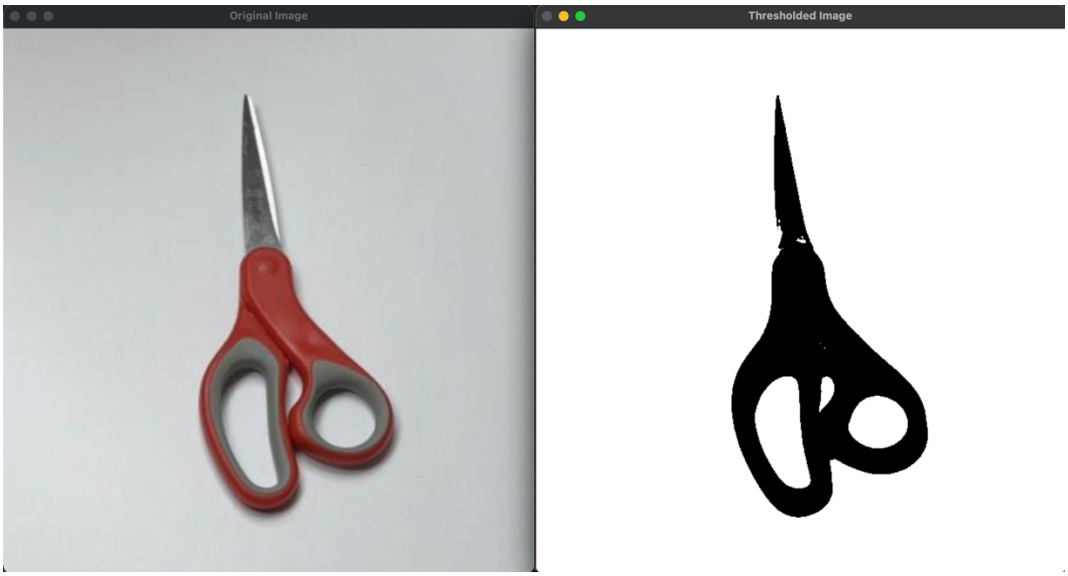
Please note that due to the real-time nature of the webcam feed, each captured image may vary slightly from the previous one, influenced by factors such as the camera's position and external lighting conditions, I use images as input.

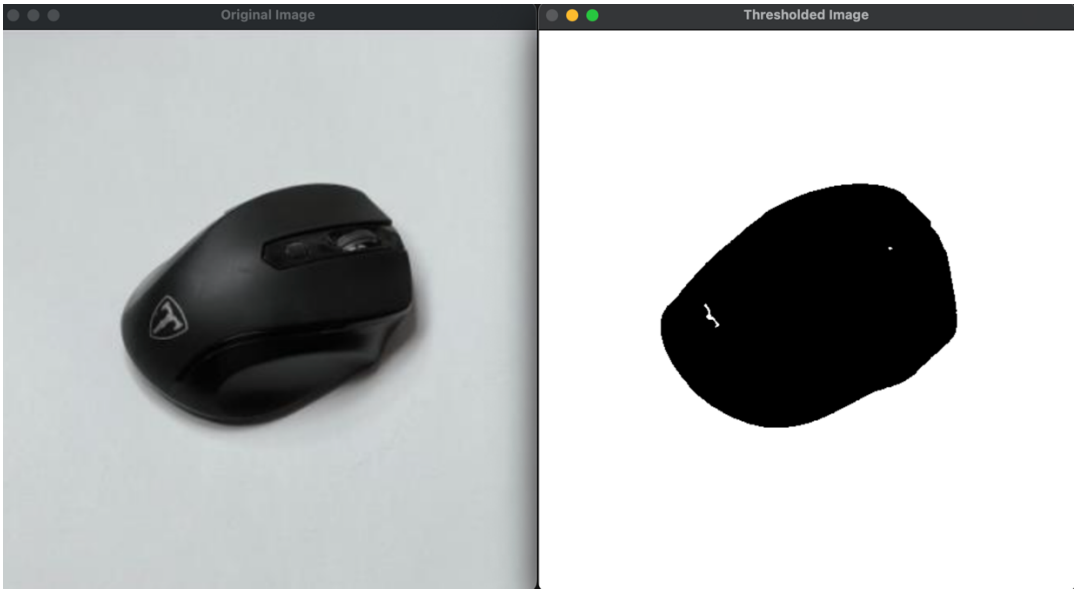
Extensions:

1. Write more of the stages of the system from scratch. (erosion and opening)
2. Experiment with more classifiers and/or distance metrics in the final task.
3. Explore some of the object recognition tools in OpenCV.

Required Image 1: Include 2-3 examples of the thresholder objects in the report.

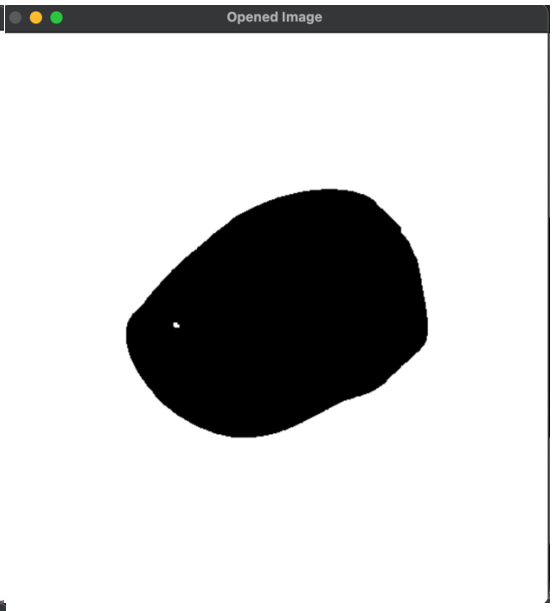
After applying thresholding to the five objects in the image, we observe that this process converts background pixels to black and foreground pixels to white, simplifying the image analysis. Additionally, K-means clustering with K=2, with a random sample of pixel values (e.g. using 1/16 of the pixels in the image) is used in this process.
Please note due to the natural light, there is some shade beside the object when taking pictures.


Original Image
Thresholded Image


Original Image
Thresholded Image

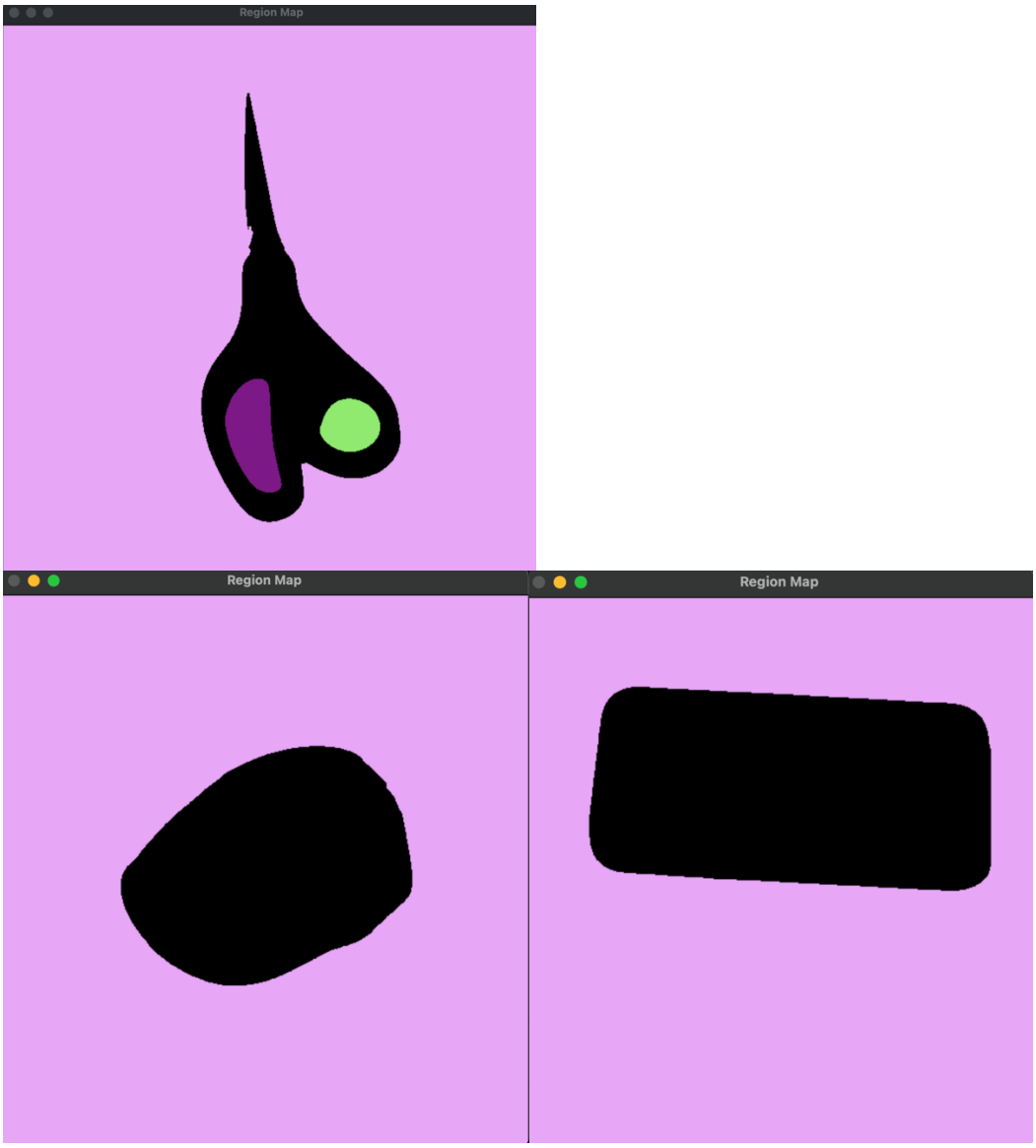

Original Image
Thresholded Image

Required 2: Include 2-3 examples of cleaned up images in your report and an explanation of your method. Use the same images as displayed for task 1.



I choose opening method to perform morphological cleaning on a binary image by first eroding and then dilating it. Erosion shrinks objects, while dilation restores their size, effectively removing small objects and noise. The process uses a defined half of the kernel size to check pixel neighborhoods and adjust the image accordingly.

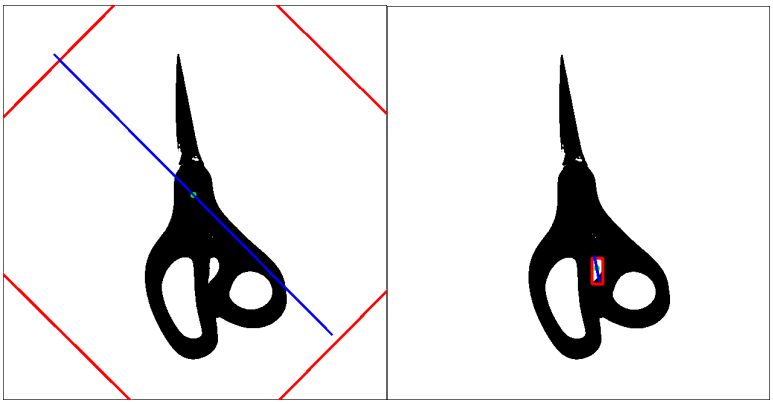
Required image 3: Include 2-3 examples of region maps, ideally with each region shown in a different color. Use the same images as for the prior tasks.



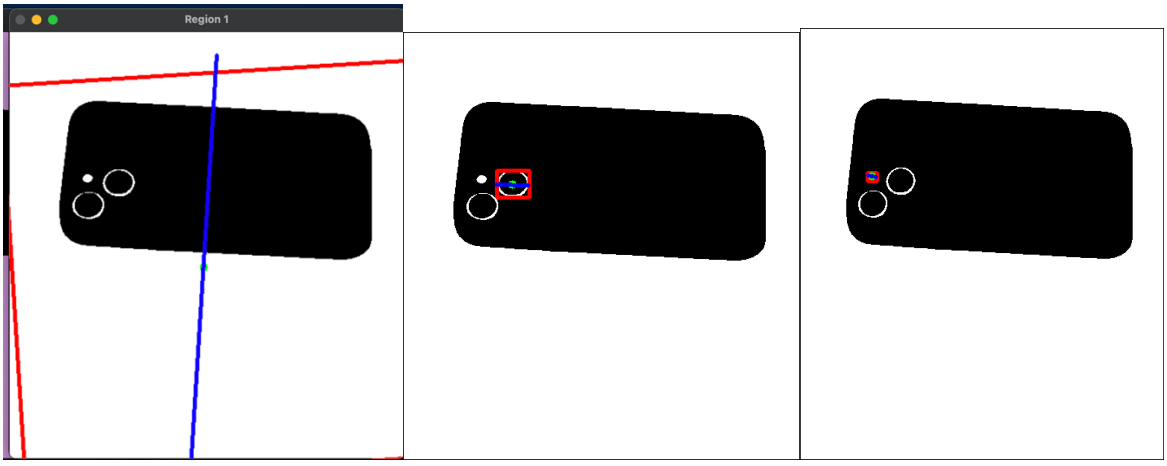
To performs connected component analysis on a binary image, I use opencv's built-in component analysis function to identify distinct regions (components) based on pixel connectivity. It assigns a unique color to each component, ensuring that regions smaller than a specified minimum size are ignored. The function returns an image where each valid component is colored randomly, providing a visual representation of the detected regions.

I added a threshold to let the region to be recognized, this is the reason that mouse and phone only show two regions.

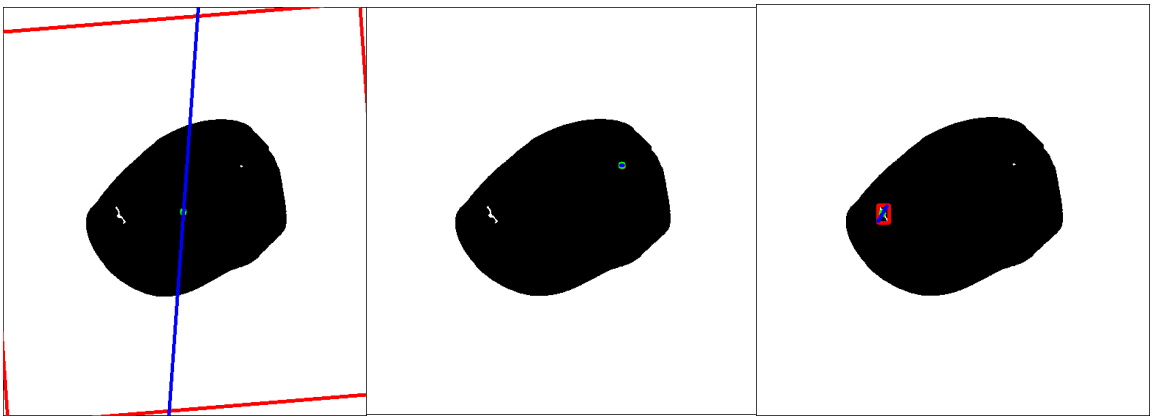
Required 4: Include 2-3 examples of regions showing the axis of least central moment and the oriented bounding box and explain your methods. Use the same images as for the prior tasks. Include the computed feature vectors for each object.



```
Region 1 features:
  Centroid: (315.085, 314.037)
  Bounding Box: 634x652
  Bounding Box Ratio: 0.972393
  Percent Filled: 0.883046
  Angle (axis of least central moment): -45.3256 degrees
Region 2 features:
  Centroid: (291.571, 231.286)
  Bounding Box: 2x4
  Bounding Box Ratio: 0.5
  Percent Filled: 0.875
  Angle (axis of least central moment): 81.6504 degrees
Region 3 features:
  Centroid: (319.2, 250.4)
  Bounding Box: 3x2
  Bounding Box Ratio: 1.5
  Percent Filled: 0.833333
  Angle (axis of least central moment): 18.4349 degrees
```



```
cathyqindeMBP:Project 3. Real-time 2-D Object Recognition cathyqin$ ./thres
Region 1 features:
  Centroid: (239.059, 290.599)
  Bounding Box: 486x524
  Bounding Box Ratio: 0.927481
  Percent Filled: 0.737411
  Angle (axis of least central moment): 86.421 degrees
Region 2 features:
  Centroid: (134.916, 187.849)
  Bounding Box: 40x33
  Bounding Box Ratio: 1.21212
  Percent Filled: 0.190152
  Angle (axis of least central moment): -2.29254 degrees
Region 3 features:
  Centroid: (96.2188, 180.552)
  Bounding Box: 13x10
  Bounding Box Ratio: 1.3
  Percent Filled: 0.738462
  Angle (axis of least central moment): -7.42477 degrees
Region 4 features:
  Centroid: (97.7644, 217.082)
  Bounding Box: 39x32
  Bounding Box Ratio: 1.21875
  Percent Filled: 0.166667
  Angle (axis of least central moment): 1.19009 degrees
Region 5 features:
  Centroid: (98, 198)
  Bounding Box: 11x1
  Bounding Box Ratio: 11
  Percent Filled: 1
  Angle (axis of least central moment): 0 degrees
```



```
Region 1 features:
  Centroid: (264.976, 279.116)
  Bounding Box: 532x556
  Bounding Box Ratio: 0.956835
  Percent Filled: 0.828163
  Angle (axis of least central moment): 85.6395 degrees
Region 2 features:
  Centroid: (349.25, 216.75)
  Bounding Box: 4x3
  Bounding Box Ratio: 1.33333
  Percent Filled: 0.666667
  Angle (axis of least central moment): 7.01812 degrees
Region 3 features:
  Centroid: (171.2, 283.933)
  Bounding Box: 15x24
  Bounding Box Ratio: 0.625
  Percent Filled: 0.208333
  Angle (axis of least central moment): 59.4059 degrees
```

To compute and visualizes various features of a region in a binary image, such as its bounding box, centroid, moments, and the axis of least central moment, I first calculate the connected components and extracts the region's statistics, such as its dimensions and area. It then computes the central moments of the region to determine the orientation of its axis and visualizes the bounding box, centroid, and the orientation line (axis of least moment) on the image, saving the result and displaying it. (centroid in green, bounding box in red, and axis orientation in blue).

Task 5: Collect training data. Explain in your report how your training system works.

I developed another entry point file training.cpp that accepts the video feed images as input and processes the images by scanning through each class and its corresponding images. Feature vectors for each image are then computed using translation, scale, and rotation-invariant metrics and stored in an object-db.csv file, which serves as the feature db.

Example:

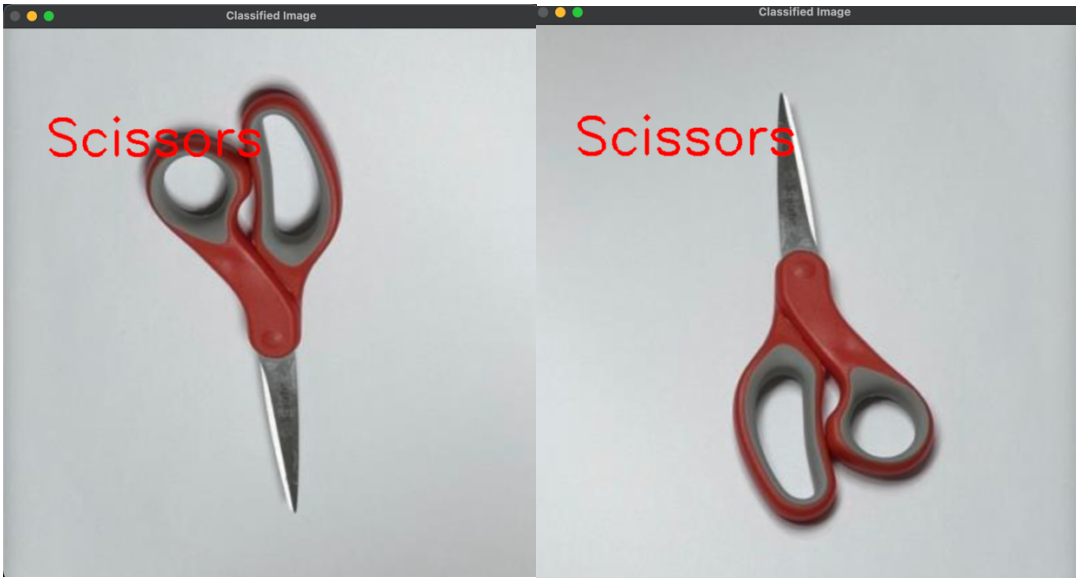
object_db.csv

```
Scissors,315.085,314.037,0.972393,0.883046
mouse,264.976,279.116,0.956835,0.828163
phone,239.059,290.599,0.927481,0.737411
headphone,134.488,161.166,0.870787,0.482947
comb,250.099,288.951,0.915194,0.841034
```

The meaning of each column is:

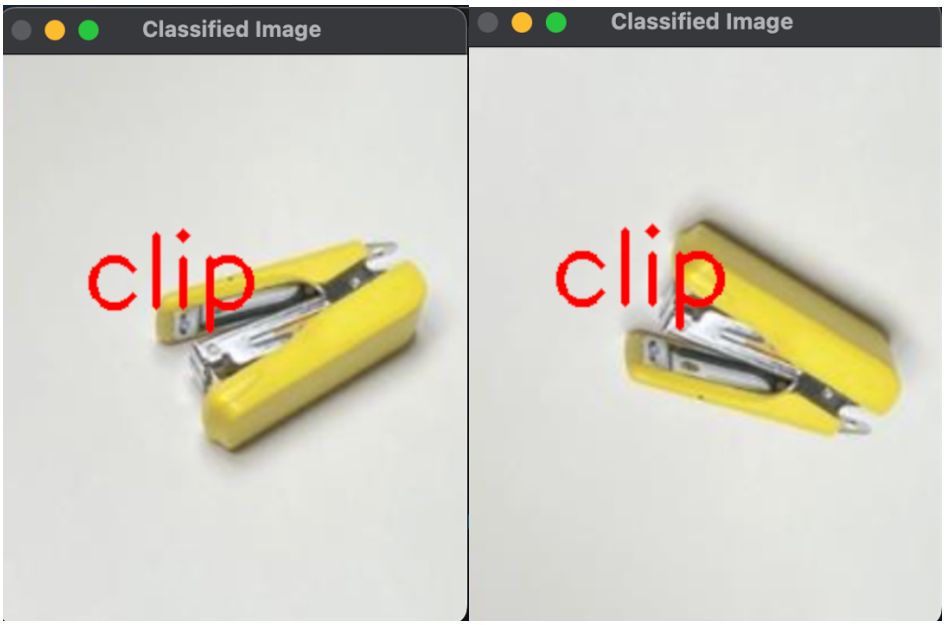
1. Label (Object Name) – The manually assigned label (e.g., "key", "phone", "scissors").
2. Centroid X (cx) – The x-coordinate of the centroid of the object.
3. Centroid Y (cy) – The y-coordinate of the centroid of the object.
4. Bounding Box Ratio (width/height) – The aspect ratio of the bounding box surrounding the object.
5. Percent Filled – The proportion of the bounding box area occupied by the actual object (object area / bounding box area).

Required image 6: include a result image for each category of object in your report with the assigned label clearly shown and explain your distance metric.




Classified Image
comb
Classified Image
comb


Classified Image
cup
cup



The distance metric used is a scaled Euclidean distance, where I first normalize the difference between corresponding features by dividing it by the standard deviation of each feature. This way features with larger values don't overpower the comparison. I then square these differences for each feature, sum them up, and take the square root to get the final distance. This method helps compare feature vectors in a way that accounts for variations in feature scales, making the comparison more accurate.

Required 7: Evaluation a 5x5 confusion matrix in your report.

I evaluated performance using test data that included three orientations per object. The accuracy was calculated using scaled Euclidean distance. Below is the confusion matrix generated from the accuracy rate results of the experiment.

name	mouse	clip	comb	cup	scissor
mouse	2/3 = 66.67%	0	0	0	0
clip	0	3 / 3 =100%	0	0	0
comb	0	0	2/3 = 66.67%	0	0
cup	0	0	0	3 / 3 =100%	0
scissor	0	0	0	0	2/3 = 66.67%

Required 8: Capture a demo of the system working.
https://drive.google.com/file/d/1xQVgOUGvn5uQ6kRlth3qM1lXgiTXWRIC/view?usp=drive_link

Required 9: Implement a second classification method

KNN:

I use OpenCV's kmeans() function to perform clustering on the feature vectors, obtaining the cluster centers and labels. Then, extract the feature vector of the new image, use kmeans() again to find the closest cluster, and return the label of the cluster corresponding to that closest center. Knn considers the labels of the K closest data points to make a prediction, typically using majority voting or averaging. In contrast, NN method uses only the closest single data point to make the prediction, without any aggregation.

name	mouse	clip	comb	cup	scissor
mouse	2/3 = 66.67%	0	0	0	0
clip	0	3 / 3 =100%	0	0	0
comb	0	0	2/3 = 66.67%	0	0
cup	0	0	0	3 / 3 =100%	0
scissor	0	0	0	0	3 / 3 =100%

Scissors set match 100%

Extension:

1. **Write more than one of the stages of the system from scratch.**

Erosion and opening (in threshold.cpp)
Erosion shrinks bright areas in a binary image by checking if all pixels in the neighborhood are white, setting the output pixel to black if any neighbor is black. Opening performs erosion followed by dilation, which helps remove noise and smooth shapes by first shrinking and then expanding the bright regions.

2. **Experiment with more classifiers and/or distance metrics in the final task.**

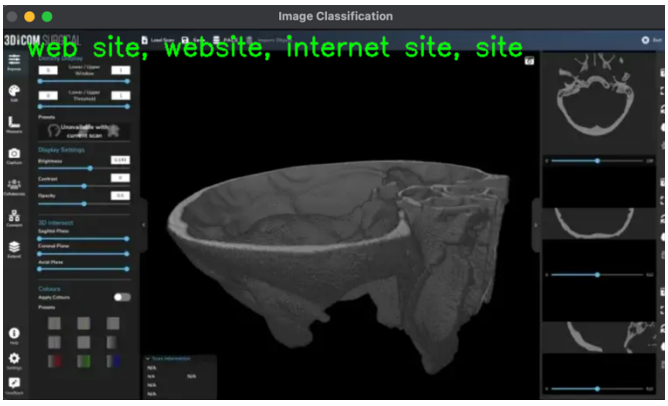
I experimented with additional classifiers and distance metrics to compare feature vectors,

Euclidean distance (total accuracy: 12/15 = 75%)
Manhattan distance (total accuracy: 7/15 = 46.67%)
Chi-square distance (total accuracy: 6/15 = 40%)

However, the performance of this distance measure was poor compared to scaled Euclidean distances, as demonstrated in the confusion matrix in task 8. The low accuracy could be attributed to several factors, such as the similar shapes of most items. Taking the number of pixels into account could enhance accuracy, and incorporating the color of objects as a feature could lead to even greater improvements.

3. **Explore some of the object recognition tools in OpenCV.**

I used a pre-trained ONNX ResNet-50 model using OpenCV's DNN module and prepares an image for classification by converting it into a blob. It then runs a forward pass through the network to obtain the class probabilities and identifies the class with the highest score. The predicted label is overlayed on the image and displayed as below:



ResNet-50 is a deep convolutional neural network with 50 layers and skip connections that help prevent vanishing gradients, making it effective for complex image classification tasks. It's commonly used for object recognition and feature extraction on large datasets, achieving high accuracy.

A short reflection of what you learned.

Through this project, I gained hands-on experience in developing a real-time object recognition system using moment features and various classifiers. I learned how to process images by applying thresholding, morphological cleaning, and connected component analysis to extract meaningful features from the objects. By experimenting with different distance metrics and classifiers, I also deepened my understanding of how feature comparison works, especially when using methods like scaled Euclidean distance for better accuracy. Additionally, integrating OpenCV's pre-trained ONNX ResNet-50 model exposed me to powerful object recognition tools and the practical application of deep learning models in computer vision tasks.

Acknowledgement:

I would like to acknowledge the resources and tutorials from OpenCV documentation, as well as guidance from professors and TA that helped me navigate through the various stages of the project.

resnet50.onnx model:

https://zenodo.org/records/2592612 https://github.com/onnx/models/tree/main