Zhimin Liang

Spring 2024 CS5330

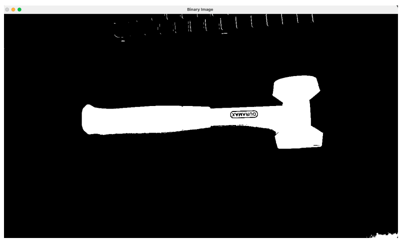
Project 3

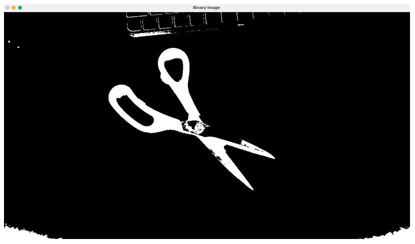
1.A short description of the overall project in your own words. (200 words or less)

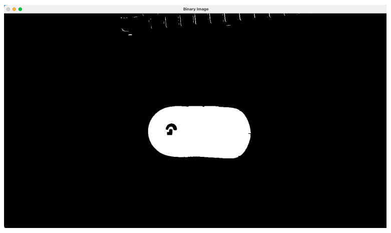
The Real-time 2D Object Recognition project aims to develop a computer system capable of identifying specified objects on a white surface in a translation, scale, and rotation invariant manner. Key tasks involve image thresholding, morphological filtering, connected components analysis, feature computation, training data collection, and implementing a classification method. Objects are chosen based on their 2D shape and uniform dark color, resembling a provided development image set. The system evaluates its performance by creating confusion matrices and includes a real-time demonstration capturing the system's functionality. Additionally, an extension involves implementing a second classification method like KNN. The project emphasizes real-time object recognition and provides flexibility for various setups, including webcam-based or video sequence processing.

2.Any required images along with a short description of the meaning of the image.

Task1: *Required images: Include 2-3 examples of the thresholded objects in your report*

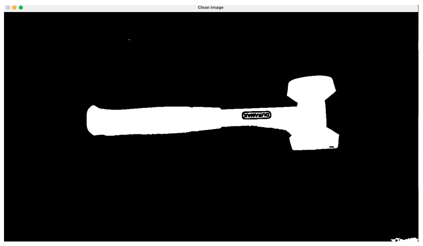


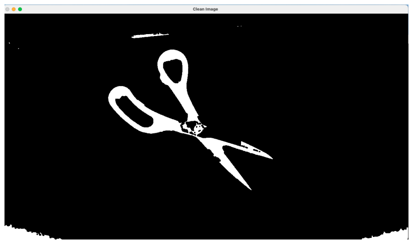


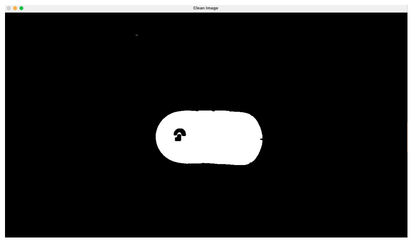


Before I pre-process the image before thresholding by blurring the image a little which can make the regions more uniform. Then I write my own code for this function which takes an input video frame (src) and converts it into a binary image (dst). The threshold value is set to 120, and the input image is first converted to grayscale. The resulting binary image has pixel values of 255 for pixels with intensity values less than or equal to the threshold, indicating foreground or object pixels. Pixels with intensity values greater than the threshold are set to 0, representing the background. The output image is essentially a black and white representation of the original image, where objects of interest are highlighted in white against a black background.

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

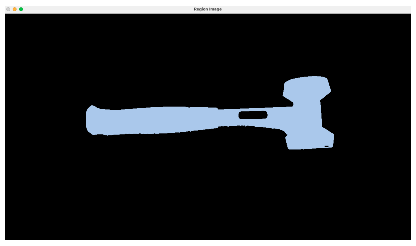


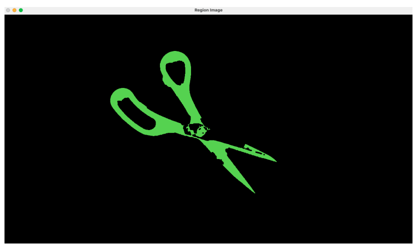


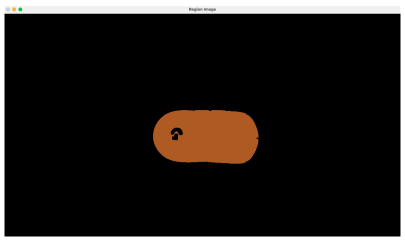


I use a median filter to clean up the binary image, since the source image in this task is just a binary pixel with 0 or 255, so I can set the threshold as 255 x 5, and then sum pixel values in the 3x3 neighborhood. if sum > threshold, it means the median is 255, so replace the center pixel with median 255(foreground), otherwise replace it as 0(background). Then I use the built- in function to shrink white regions. Therefore, we can find that some noise at the top of the three binary images have been removed.

Task 3: Required images: 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



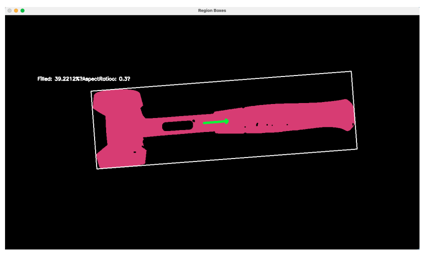




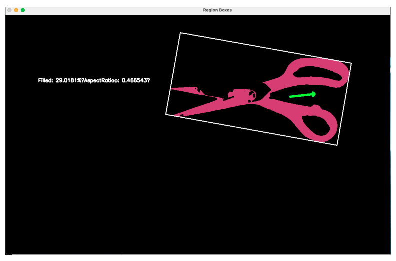
I use function segmentation to take a cleaned binary image (cleanImage) and perform connected components labeling to identify distinct regions.I set minRegionSize to ignore any regions that are smaller than it, so we can found that the bottom white region at the bottom in the clean image have been ignored.

Then I assign a consistent color to each recognized region. The output includes the region label map (regionLabelMap), statistical information about each region (stats), and the centroids of the regions (centroids). The colored region map is returned.

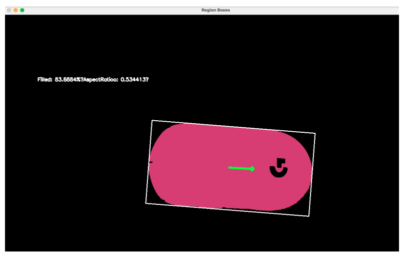
Task 4:Required images: Include 2-3 examples of regions showing the axis of least central moment and the oriented bounding box. use the same images as for the prior tasks. Include the computed feature vectors for each object.



```
Filled: 39.2212%?AspectRatioo: 0.3?
```



```
Filled: 29.0181%?AspectRatioo: 0.466543?
```



```
Filled: 83.6884%?AspectRatioo: 0.534413?
```

From the images we can find the regions showing the axis of least central moment and the oriented bounding box. And the % filled and AspectRatio features are shown on the image.

The function computes the moments for each region using the OpenCV cv::moments function. Moments are statistical measures used to describe the distribution of pixel intensities within a region. The moments include:

- m00 (Zeroth Order Moment): Represents the total intensity or area of the region.
- m01, m10 (First Order Moments): Used to calculate the centroid of the region.
- mu11, mu20, mu02 (Central Moments): Used to calculate the orientation of the region.

Feature Vector:

1.Centroid: As explained, this is the geometric center of the shape. It's useful for tracking the position of objects.

2.Orientation: This feature indicates how the object is oriented with respect to the coordinate system or its axis of least inertia. It can be critical for recognizing objects that have a specific orientation.

3.Percent Filled: This is calculated as the ratio of the area of the object to the area of its bounding box, multiplied by 100 to get a percentage. It gives an idea of how much of the bounding box is occupied by the object, which can be a useful feature for distinguishing between objects of similar shapes but different densities or fill ratios.

4.Aspect Ratio: Calculated as the ratio of the height to the width of the bounding box of the object. This feature helps in distinguishing objects based on their elongation.

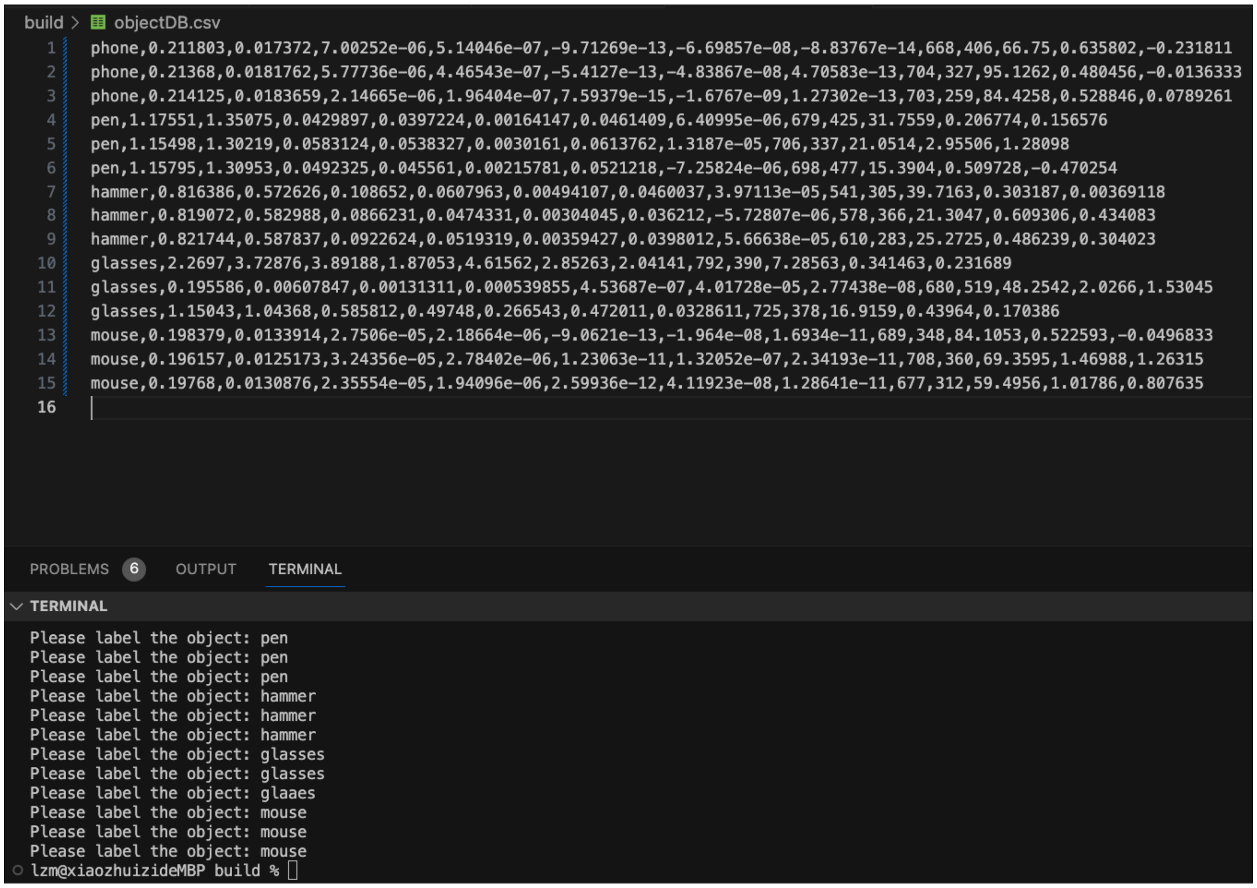
5. huMoments: calculate the Hu1, Hu2, ..Hu7 Hu Moment based on moments.

Task 5: Explain in your report how your training systems works.Explain in your report how your training systems works.

Once segmentation is complete, each identified region is labeled, and properties such as area, bounding box, and centroids are calculated. I start to do the data collection by the following two step:

- Labeling: The user is prompted to input a label for the object. This label is what the feature vector will be associated with, acting as the target output for the training data.
- Saving Features: The extracted features, along with the label, are saved to a objextDB.csv file. Each line in the file begins with the label, followed by the feature vector, with each feature separated by commas. This structured format allows for easy parsing and use in comparing for the new feature in the next task.

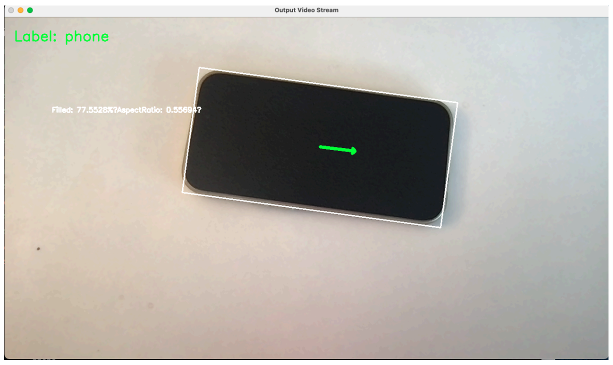
I collect each 5 objects in different positions and orientations



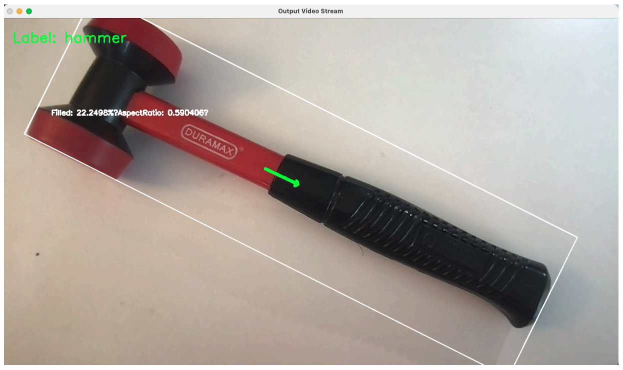
```
phone,0.211803,0.017372,7.00252e-06,5.14046e-07,-9.71269e-13,-6.69857e-08,-8.83767e-14,668,406,66.75,0.635802,-0.231811
phone,0.21368,0.0181762,5.77736e-06,4.46543e-07,-5.4127e-13,-4.83867e-08,4.70583e-13,704,327,95.1262,0.480456,-0.0136333
phone,0.214125,0.0183659,2.14665e-06,1.96404e-07,7.59379e-15,-1.6767e-09,1.27302e-13,703,259,84.4258,0.528846,0.0789261
pen,1.17551,1.35075,0.0429897,0.0397224,0.00164147,0.0461409,6.40995e-06,679,425,31.7559,0.206774,0.156576
pen,1.15498,1.30219,0.0583124,0.0538327,0.0030161,0.0613762,1.3187e-05,706,337,21.0514,2.95506,1.28098
pen,1.15795,1.30953,0.0492325,0.045561,0.00215781,0.0521218,-7.25824e-06,698,477,15.3904,0.509728,-0.470254
hammer,0.816386,0.572626,0.108652,0.0607963,0.00494107,0.0460037,3.97113e-05,541,305,39.7163,0.303187,0.00369118
hammer,0.819072,0.582988,0.0866231,0.0474331,0.00304045,0.036212,-5.72807e-06,578,366,21.3047,0.609306,0.434083
hammer,0.821744,0.587837,0.0922624,0.0519319,0.00359427,0.0398012,5.66638e-05,610,283,25.2725,0.486239,0.304023
glasses,2.2697,3.72876,3.89188,1.87053,4.61562,2.85263,2.04141,792,390,7.28563,0.341463,0.231689
glasses,0.195586,0.00607847,0.00131311,0.000539855,4.53687e-07,4.01728e-05,2.77438e-08,680,519,48.2542,2.0266,1.53045
glasses,1.15043,1.04368,0.585812,0.49748,0.266543,0.472011,0.0328611,725,378,16.9159,0.43964,0.170386
mouse,0.198379,0.0133914,2.7506e-05,2.18664e-06,-9.0621e-13,-1.964e-08,1.6934e-11,689,348,84.1053,0.522593,-0.0496833
mouse,0.196157,0.0125173,3.24356e-05,2.78402e-06,1.23063e-11,1.32052e-07,2.34193e-11,708,360,69.3595,1.46988,1.26315
mouse,0.19768,0.0130876,2.35554e-05,1.94096e-06,2.59936e-12,4.11923e-08,1.28641e-11,677,312,59.4956,1.01786,0.807635
```

```
Please label the object: pen
Please label the object: pen
Please label the object: pen
Please label the object: hammer
Please label the object: hammer
Please label the object: hammer
Please label the object: glasses
Please label the object: glasses
Please label the object: glaaes
Please label the object: mouse
Please label the object: mouse
Please label the object: mouse
lzm@xiaozhuizideMBP build %
```

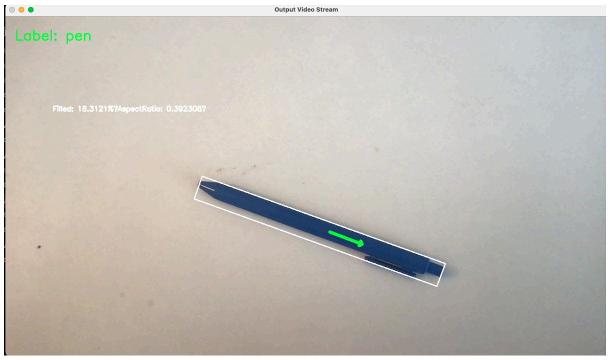
Task 6: Required images: include a result image for each category of object in your report with the assigned label clearly shown.



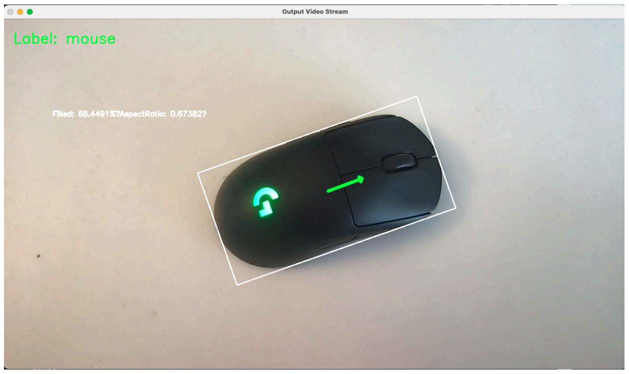
```
Label: phone
Filled: 77.5528%?AspectRatio: 0.55694?
```



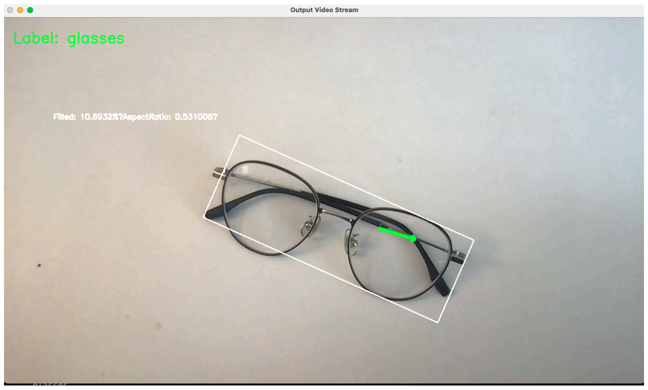
```
Label: hammer
Filled: 22.2498%?AspectRatio: 0.590406?
```



```
Label: pen
Filled: 18.3121%?AspectRatio: 0.392308?
```



```
Label: mouse
Filled: 68.4491%?AspectRatio: 0.67382?
```



```
Label: glasses
Filled: 10.6932%?AspectRatio: 0.531008?
```

I collect 5 categories in my database which are pen, mouse, hammer, phone, and glasses. And the images show that my system is able to show the bounding box and central moment and then label the object based on the closest distance with the known object in the database.

Task 7.Evaluate the performance of your system on at least 3 different images of each object in different positions and orientations. Build a 5x5 confusion matrix of the results showing true labels versus classified labels. Include the confusion matrix in your report.

	pen	mouse	phone	hammer	glasses
pen	***3***	***0***	***0***	***1***	***0***
mouse	***0***	***3***	***2***	***0***	***0***
phone	***0***	***1***	***3***	***0***	***0***
hammer	***0***	***0***	***1***	***3***	***0***
glasses	***1***	***0***	***0***	***0***	***3***

Based on the matrix, the phone has been classified as a mouse or the mouse being recognized as a phone, I think that is because when I collect data in the database, I use the mouse and phone to have similar size which will lead to the %filled, and aspect Ratio similar.

Task 8: Capture a demo of your system working

https://youtu.be/vAmdNZNI7Ms

Task 9: Explain your choice and how you implemented it. Also, compare the performance with your baseline system and include the comparison in your report.

I chose to implement K-Nearest Neighbors (KNN) as a second classification method due to its simplicity and effectiveness in handling classification tasks, especially in scenarios where the decision boundary is complex. KNN is a non-parametric and instance-based learning algorithm, making it suitable for our object recognition problem.I converted the known objects' feature vectors into training samples, similar to the baseline KNN implementation.I used the cv::ml::KNearest class from the OpenCV library and set K > 1 to find multiple nearest neighbors during classification.

3. A description and example images of any extensions.

I write my own code for both Threshold and Clean up

4.A short reflection of what you learned.

In completing the Real-time 2D Object Recognition project, I gained valuable insights into the intricacies of image processing, feature extraction, and classification techniques. The project provided hands-on experience in implementing fundamental computer vision tasks, such as thresholding, morphological filtering, and connected components analysis, to preprocess and segment images effectively. Understanding the importance of translation, scale, and rotation invariance in feature computation was a key takeaway, guiding the development of robust object recognition.The implementation of a training mode for collecting feature vectors and building an object database added a practical dimension to the project, demonstrating the significance of labeled data in machine learning applications.

5.Acknowledgement of any materials or people you consulted for the assignment.

- **TAs:** Tejaswini Dilip Deore, Sasank Potluri
- **OpenCV Tutorials:** https://docs.opencv.org/4.5.1/d9/df8/tutorial_root.html
- **YouTube:**

  Computer Vision Lab, “OPENCV & C++ TUTORIAL”,https://www.youtube.com/playlist?list=PLUTbi0GOQwghR9db9p6yHqwvzc989q_mu [access, Feb,29,2024]
- **Stackoverflow**:
  Stackoverflow, “Extracting the information of angle and axis ”https://stackoverflow.com/questions/56172504/extracting-the-information-of-angle-and-axis [access, Feb,29,2024]

  Stackoverflow, “OpenCV 3 KNN implementation, ”https://stackoverflow.com/questions/28035484/opencv-3-knn-implementation [access, Feb,29,2024]