Leaksmy Heng

CS5330

Project3

Feb 21, 2025

Report

Short Description of Overall Project

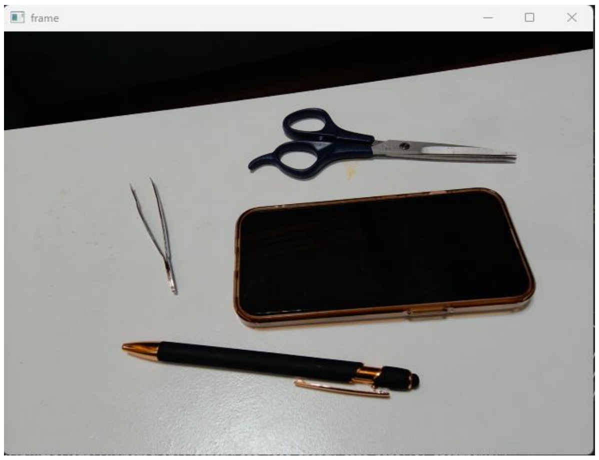
This project is interesting and challenging. This is by far the most challenging one out of the tree. However, I am quite excited working on it especially the inferencing part where I had the object in front of the camera and had it trying to identify the object. I had fun in that part as I put a peace sign in front of the camera and sometimes it thinks of it as scissor. If I have more time, I'll train more data with different positions. I think with more data on different positions, it will pick up the object correctly. Overall, it is a fun and challenging project that I enjoyed a lot.

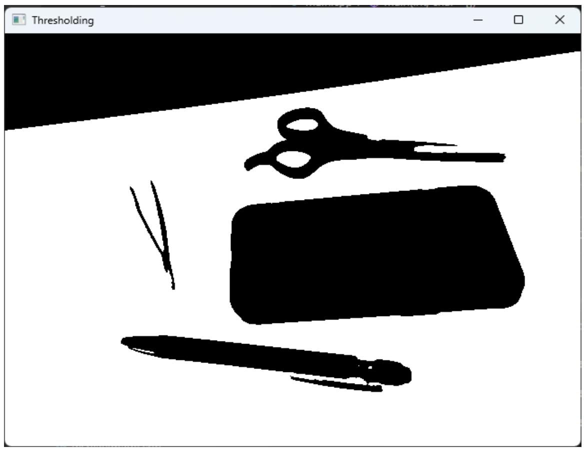
Short Description of What I Have Learned

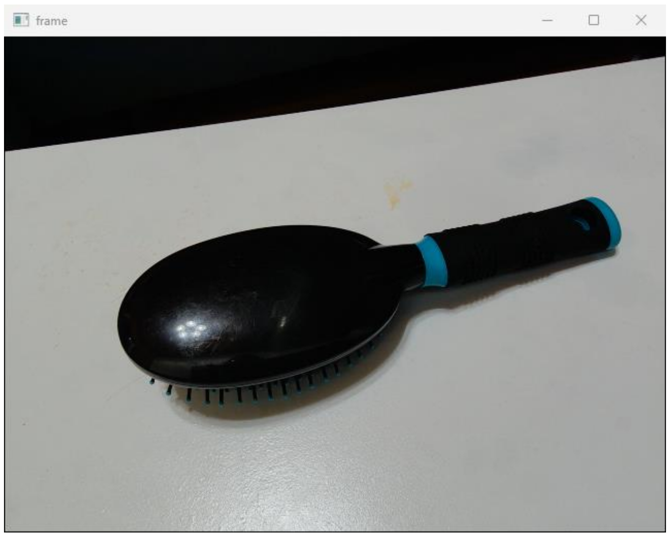
This project has been a great learning experience for me. It started with extracting features from video frames, then having the user label these extractions to build a training dataset. Using the labeled data, I was able to apply it to classify objects in real-time. Of course, the results are not perfect because of the limited size of the training set, but it has definitely inspired me to explore this further and improve the accuracy of the system. I'm excited to continue developing and refining this approach in future projects.

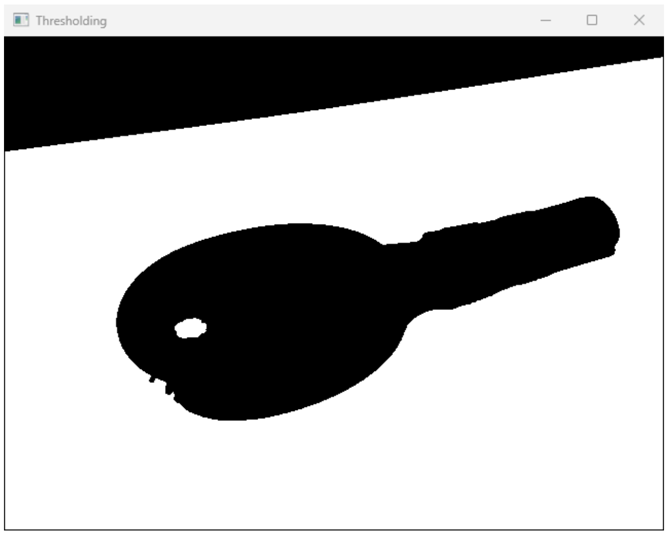
Task1: Threshold the input video

I implemented thresholding by first using gaussian filter 5x5 to smooth the image. After that, I used OpenCV to convert the frame to grayscale frame then applied threshold function to it. 127 is the threshold value; therefore, any pixel above that will be marked as 255 and any pixel below or equal to that will be marked as 0.



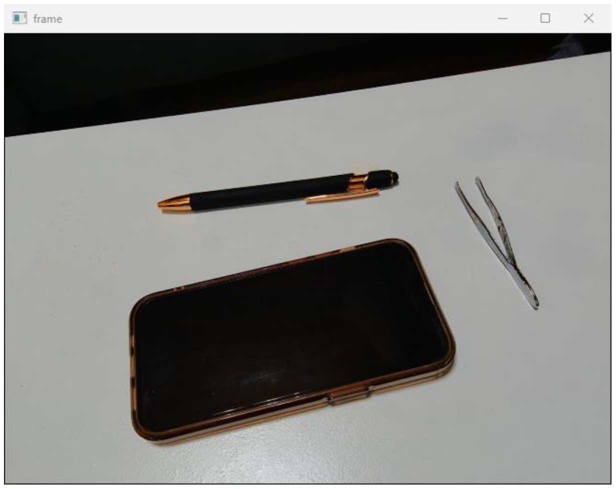


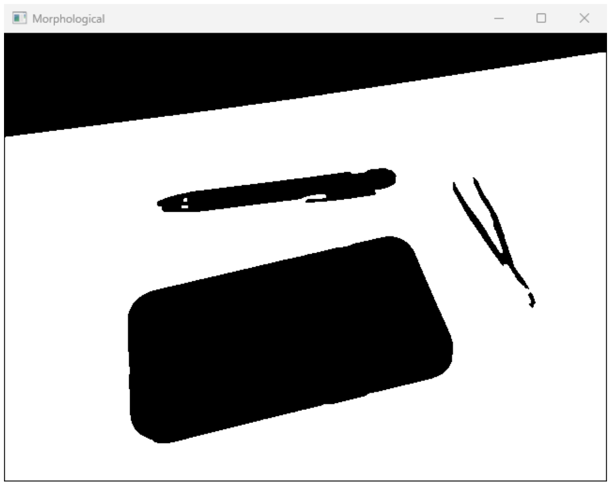


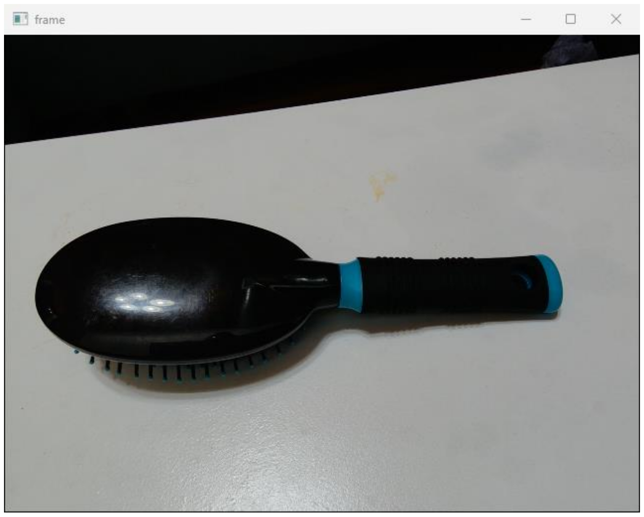


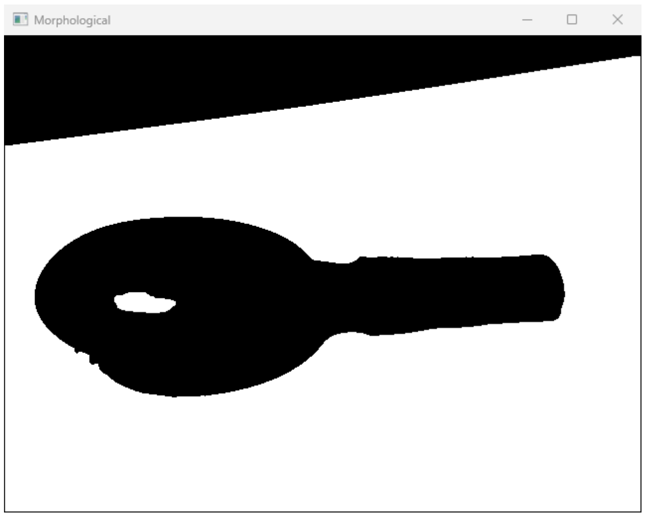
Task2: Clean up the binary image

I applied open morphological filtering onto the binary image above. In OpenCV, there is a function for open filtering directly; however, I didn't use it. I used the cv::erode and cv::dilation because I want to customize the kernel used in those two functions. For erosion, I used 4 connected kernels and for dilation, I used 8 connected kernels.





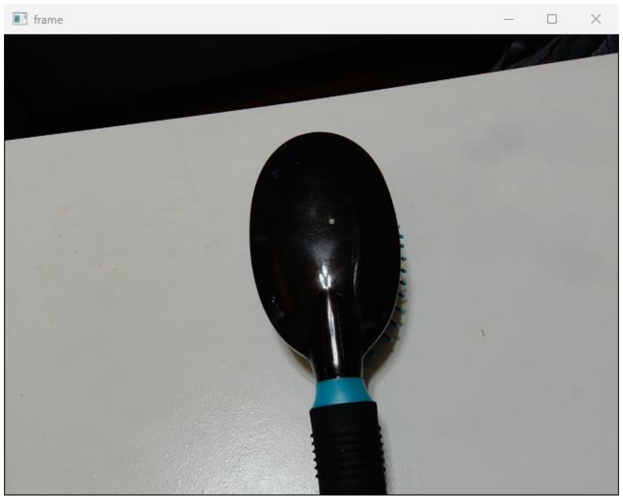


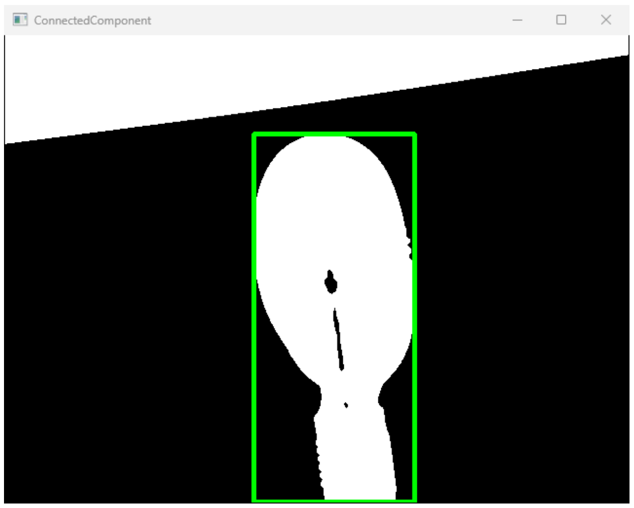


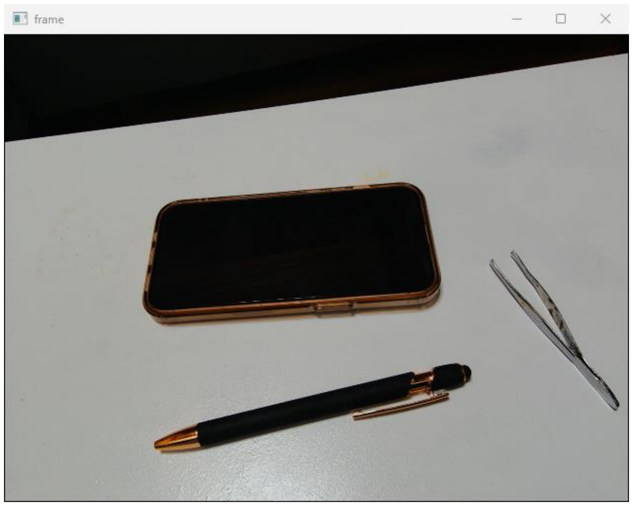
Task3: Segment the image into regions

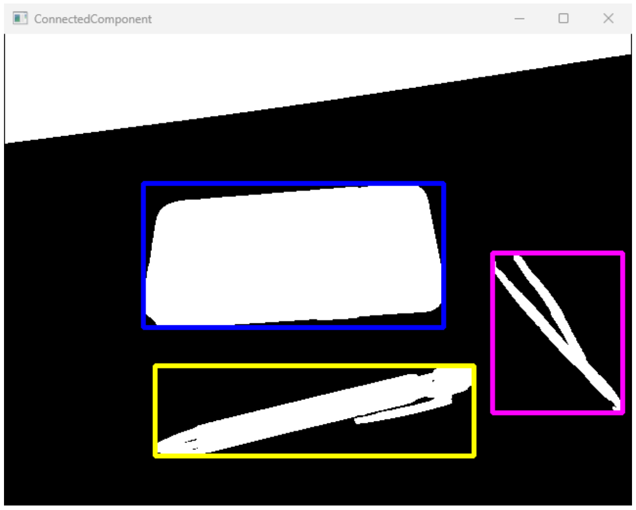
This task is interesting, and I have spent a lot of time here trying to understand why my video is not being segmented correctly. Two issues I have encountered:

- The object was not being segmented correctly. It picked some random noises like a small white spot in between the black object. It turned out the function I used to for thresholding
  "cv::threshold(frame, grayFrame, 125, 255, cv::THRESH_BINARY);"
  was inverting the objects. I want the object to be white and the background to be black.
  After changing the function to
  "cv::threshold(frame, grayFrame, 125, 255, cv:: THRESH_BINARY_INV);"
  The segmentation is working correctly.
- The second issue was the rectangle used to segment the objects are all in the same color even though I have used random color generator to generate different color. I tried multiple solutions by first having the seed function srand in main, or in the for loop. Even increment the seed by one in each iteration to multiply it with the random generator to get different results, but my rectangle box is still in white. I think this is because this is in a video so time(0) does not change much, but I could be wrong. Now I just have a fix 5 color sent and have it pick out the color randomly out of the five for each of the region and it works.



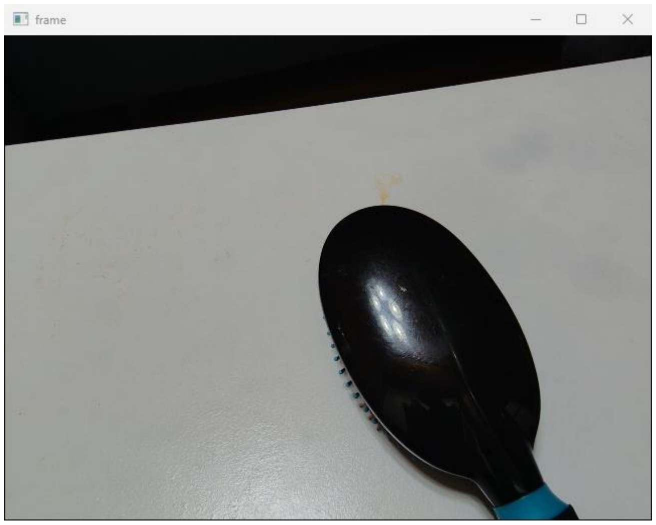


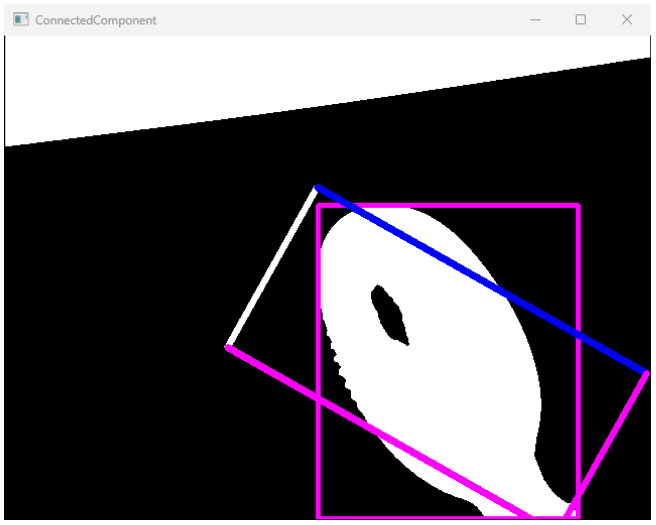


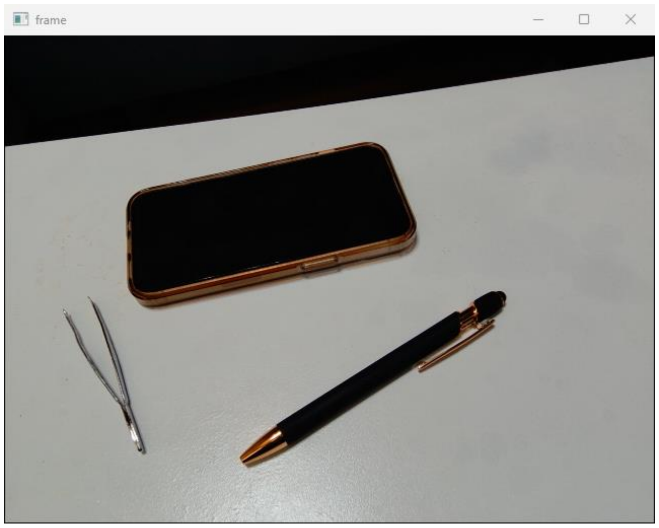


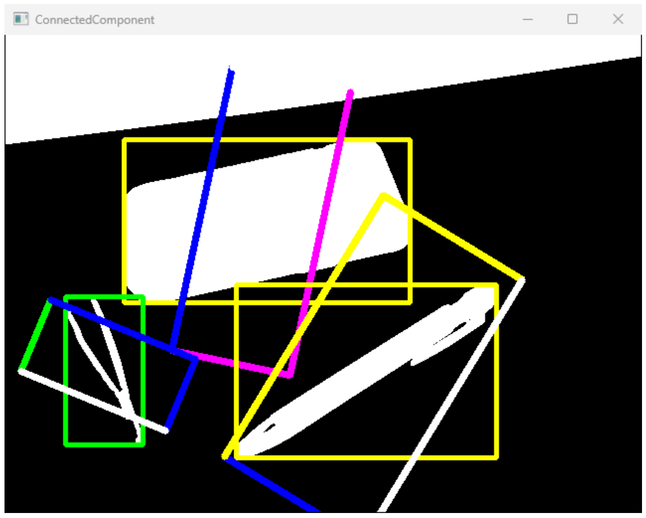
Task4: Compute features for each major region

For task 4, I used the stats result from the previous task as an input and was able to calculate some features including axis of least central moment, percent filled, and bounding box height/width ratio. However, when I tried to draw the bounding box by first getting the rotated rectangle using cv::RotatedRect obb = cv::minAreaRect(region); my code kept on throwing error regarding to contouring. I have spent about 1 day trying to figure this bug out, but I still could not solve it. I found code online from a previous student of this course, so I used the code in this part here. I have put out their github where I found their code in the reference part along with in the documentation of the code. This is the only code of this entire project that I borrowed from their repository.









Task5: Collect training data

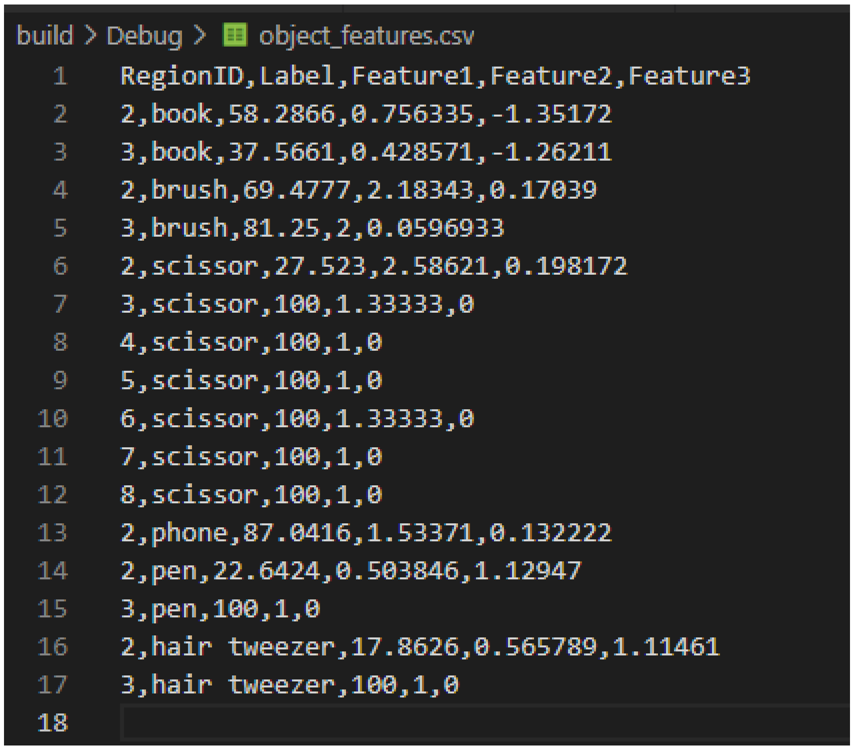
I implemented this part by

- having the program listen to the key stroke.
- If the user type in n or N, the boolean for whether or not to save the file will turn True and the program will ask for the user input in a text saying "Enter a label for the current object:".

- Once users finish label the object and press ENTER, the label, region_id, and features including including axis of least central moment, percent filled, and bounding box height/width ratio will get store in and ObjectFeature structure which then push to the vector.
- Once finish label all the test objects, users can type in s or S, the data in the vector will be read and write to a csv file.
- In case users forgot to save and just press q or Q to quite, the program will save that for them too.

I did not have the program write to csv directly one object at a time because when I implemented it, the video become lagging; therefore, it's not really user-friendly.

Output in csv file



```
build > Debug > object_features.csv
RegionID,Label,Feature1,Feature2,Feature3
2,book,58.2866,0.756335,-1.35172
3,book,37.5661,0.428571,-1.26211
2,brush,69.4777,2.18343,0.17039
3,brush,81.25,2,0.0596933
2,scissor,27.523,2.58621,0.198172
3,scissor,100,1.33333,0
4,scissor,100,1,0
5,scissor,100,1,0
6,scissor,100,1.33333,0
7,scissor,100,1,0
8,scissor,100,1,0
2,phone,87.0416,1.53371,0.132222
2,pen,22.6424,0.503846,1.12947
3,pen,100,1,0
2,hair tweezer,17.8626,0.565789,1.11461
3,hair tweezer,100,1,0
```

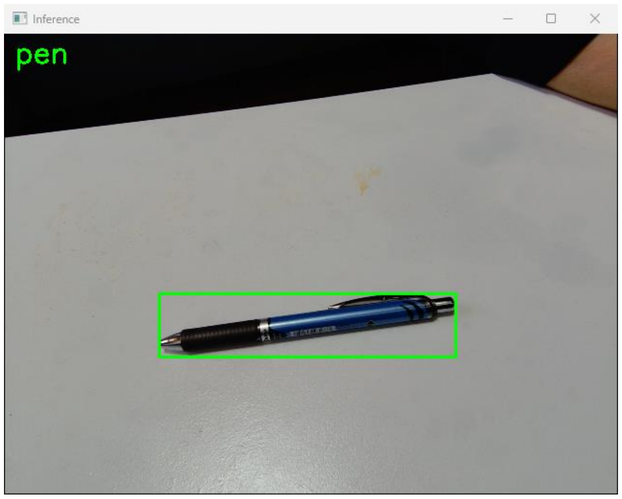
Task6: Classify new images

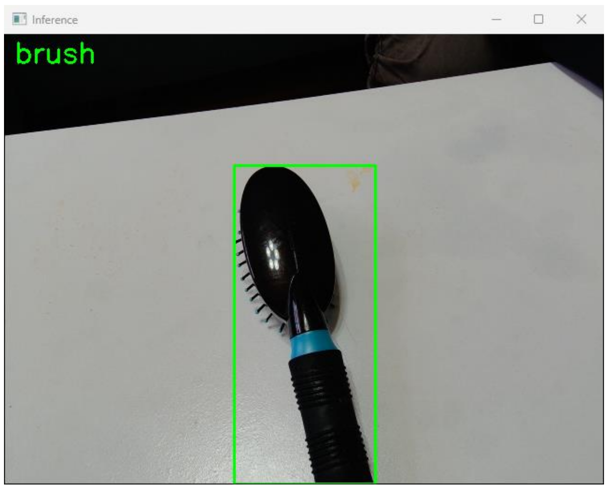
For 6 is to classify the new image, if the system does not recognize the image, show the object is UNKNOWN. With limited data as shown above and limit position of each object, I found my program to be not good with classifying the object when I moved the position of the object.

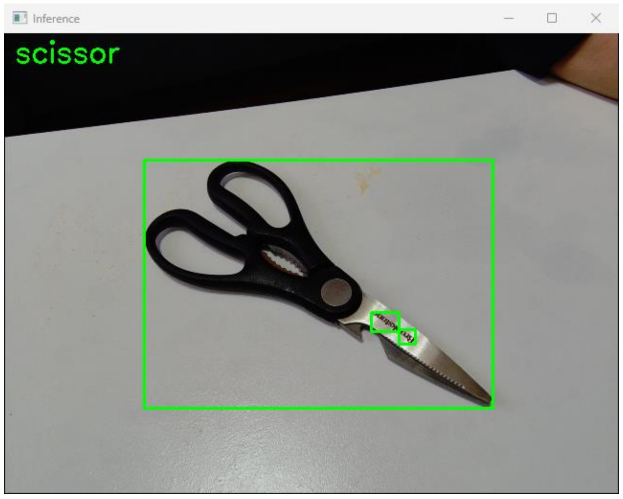
To classify the object, I used scaled Euclidean distance metric. Here is the step-by-step implementation:

- Read the training feature vector from csv and store it in a struct vector
- Calculate the standard deviation of the training feature vector

- Find the appropriate threshold using 3-sigma rule (99.7% of the data falls within three standard deviations of the mean in a normal distribution)
- Extract the feature from the current frame.
- Classify the object in the current frame using the Euclidean distance metric. If the Euclidean distance is smaller than the current minDistance (maximum possible double value), it updates minDistance and sets the label to the known object's label. Otherwise, check if the distance is greater than the threshold set above, then mark the object as unknown.

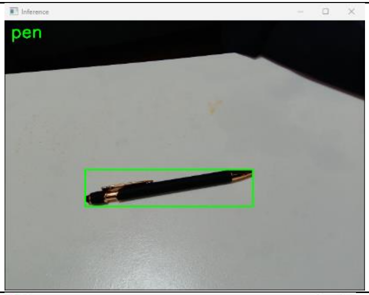
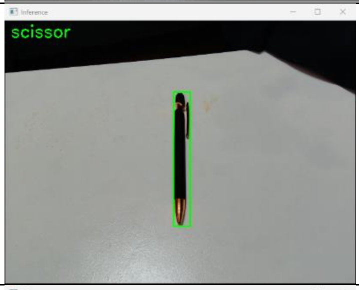
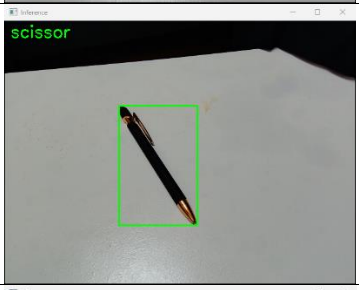
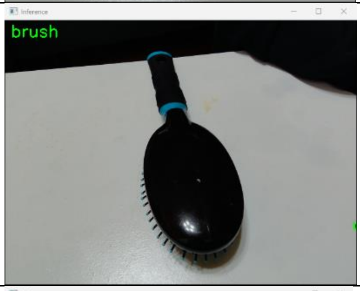
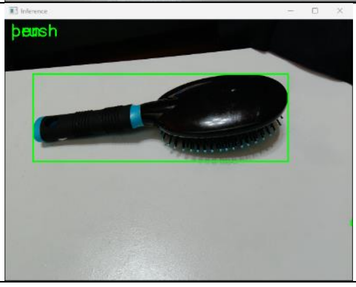


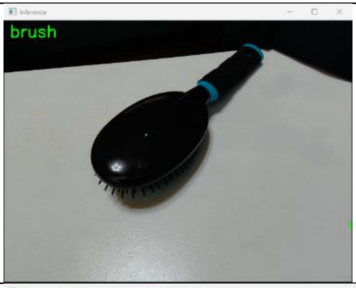
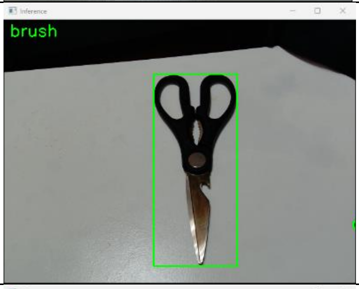
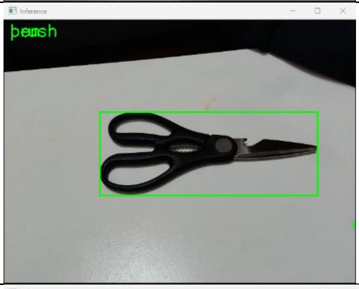
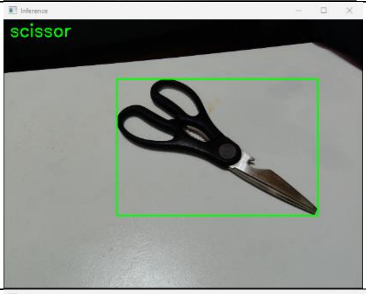
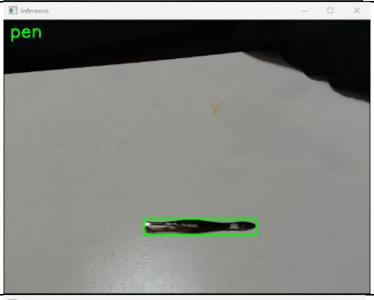
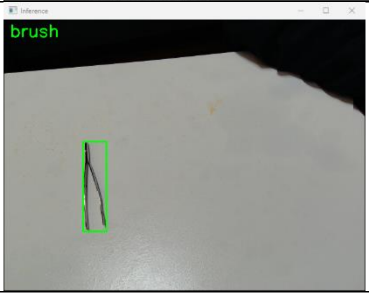


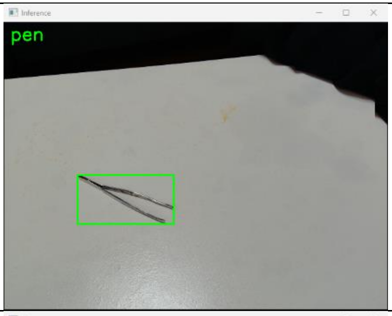
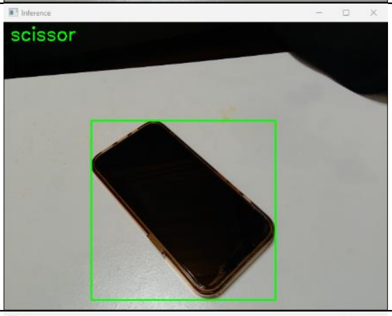
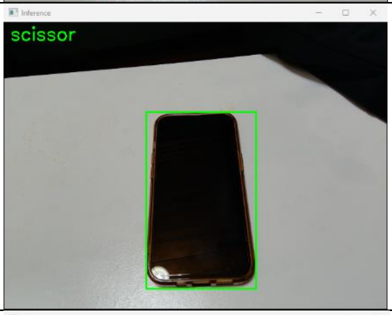
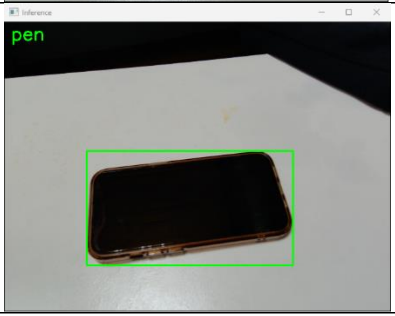


Task7: Evaluate the performance of your system

Data that I used to evaluate my performance.

Image	Different Position	Predicted	Actual
Pen		**Pen**	**Pen**
Pen		**Scissor**	**Pen**
Pen		**Scissor**	**Pen**
Brush		**Brush**	**Brush**
Brush		**Brush**	**Brush**

Brush		**Brush**	**Brush**
Scissor		**Brush**	**Scissor**
Scissor		**Brush**	**Scissor**
Scissor		**Scissor**	**Scissor**
hair tweezer		**pen**	**hair tweezer**
hair tweezer		**pen**	**hair tweezer**

hair tweezer		**pen**	**hair tweezer**
phone		**scissor**	**phone**
phone		**scissor**	**phone**
phone		**pen**	**phone**

Confusion metric

Predicted (right) / Actual (down)	**Pen**	**Brush**	**Scissor**	**Hair Tweezer**	**Phone**	**Total**
Pen	1	0	2	0	0	**3**
Brush	0	3	0	0	0	**3**
Scissor	0	2	1	0	0	**3**
Hair Tweezer	3	0	0	0	0	**3**
Phone	1	0	2	0	0	**3**
Total	**5**	**5**	**5**	**0**	**0**	**15**

Task8: Capture a demo of your system working

Link to video: https://drive.google.com/file/d/1zmXUUSit1t403kYVoDV-yO-WCWPcOL5W/view?usp=drive_link

Task9: Implement a second classification method

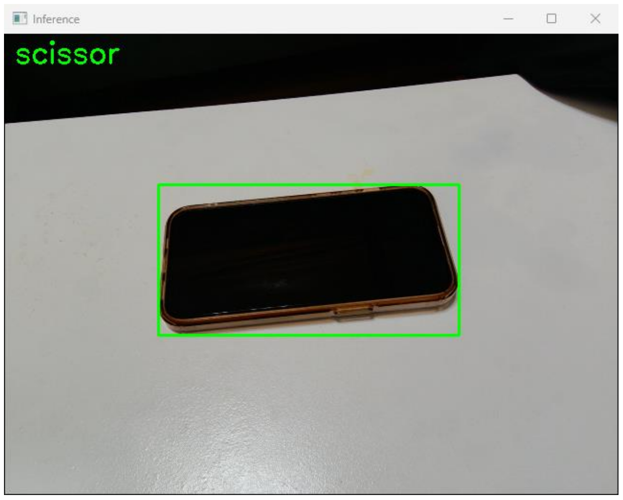
For this task, I implemented KNN. This is how implemented KNN without using voting method:

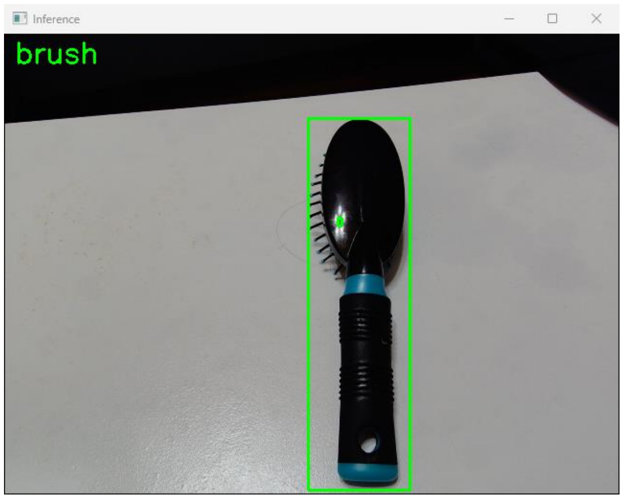
- Calculate the distance between features using standard Euclidean distance.
- Sort the distance result in ascending order
- Compute the average feature vector of the K nearest neighbors
- Average the features
- Find the label of the nearest neighbor whose feature vector is closest to the average feature vector

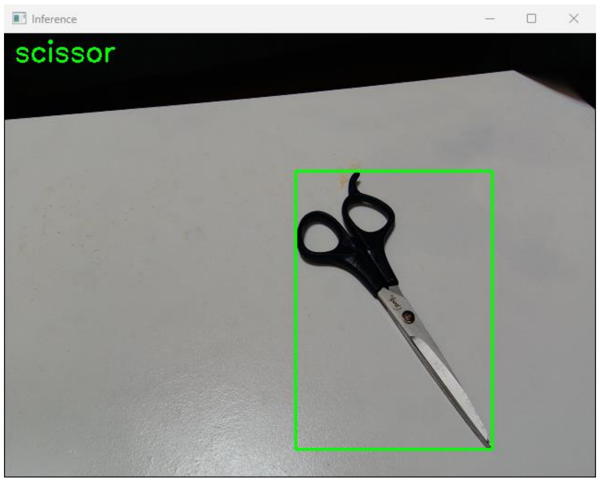
I also did some experiments here:

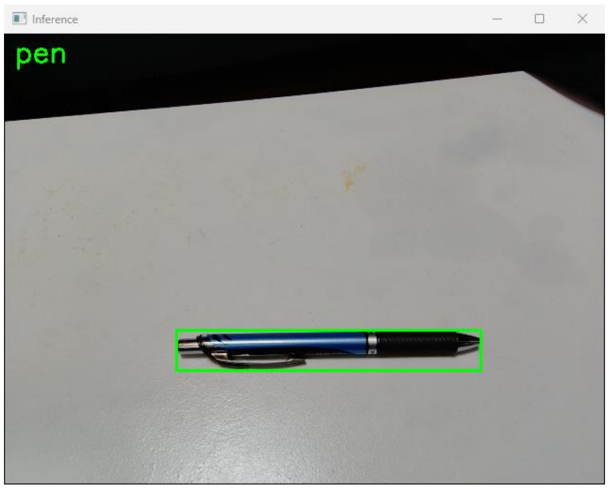
First, I only use one feature at a time using absolute distance value (abs(f1 – f2)). I found out that using just one feature is not as good as using all of it. For example, when I use just a percent fill feature, almost everything becomes tweezer. When I change it to height/width ratio, it was able to identify book and brushes, but the rest of the object were identical incorrectly.

At the end, I opted out to use all features with standard Euclidean distance. The result is still the same as the other method. It was not able to classify the phone correctly. But for other object, it is slightly better.





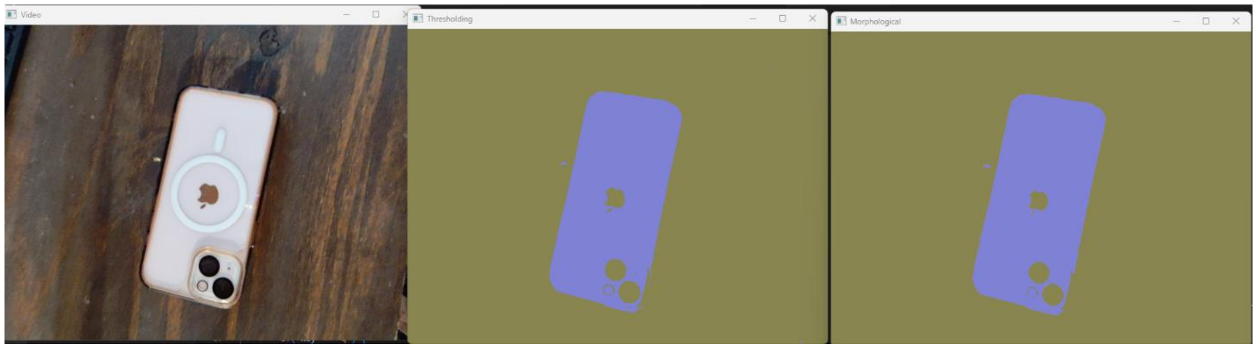


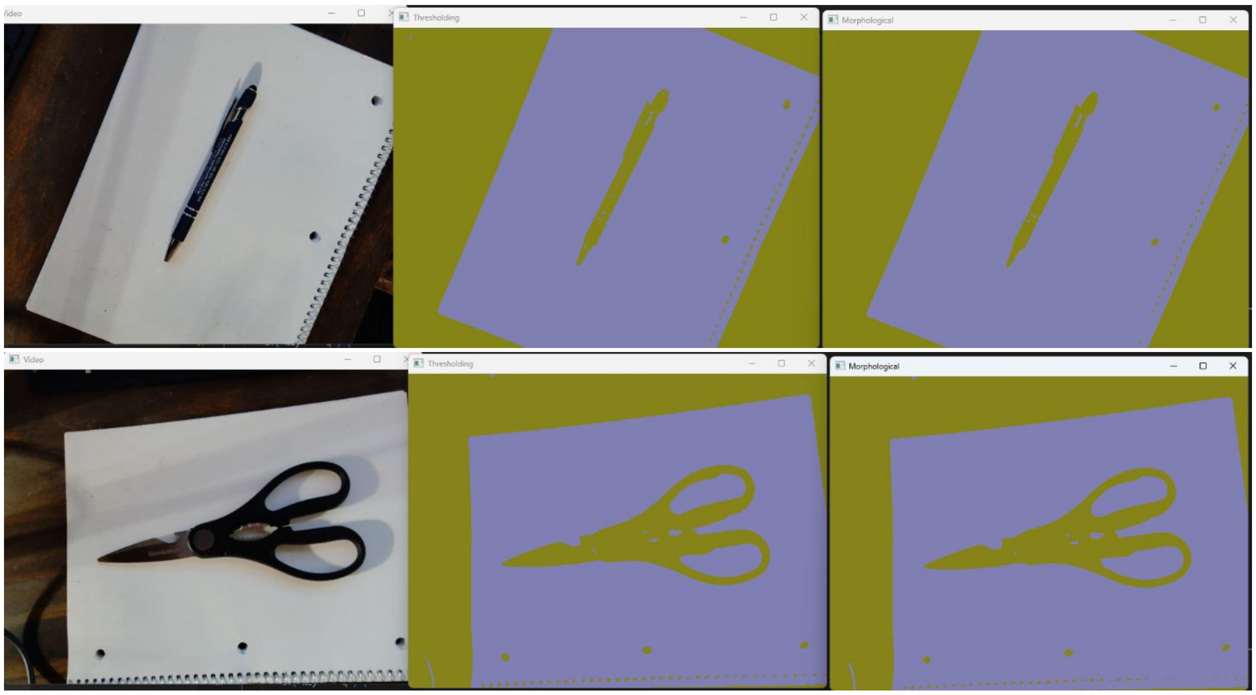


Extension:

Extension1:

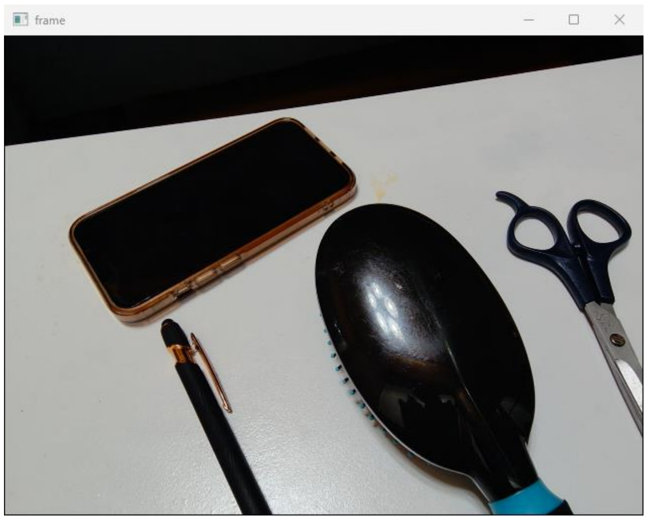
For extension, in task 1, I applied the k-mean cluster to convert the video to binary color using k=2. However, k-mean cluster causes lagging to the video. Therefore, throughout the project, I decided to use regular threshold algorithms instead. Below is the image using k-mean before I uses binary black and white onto it.

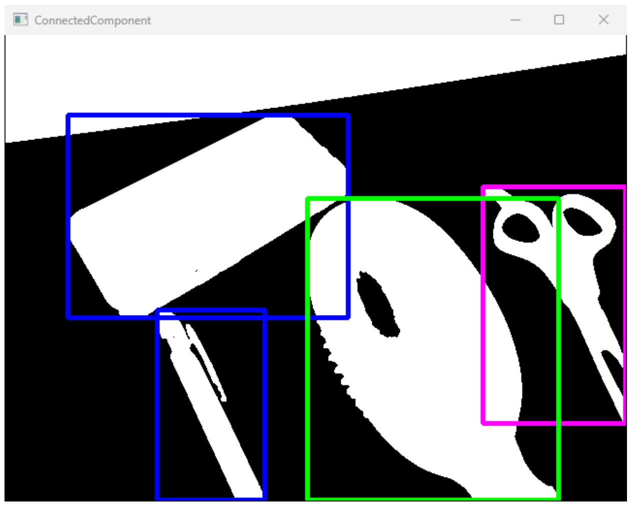




Extension 2:

As seen on task 3, my program was able to pick up multiple objects at the same time.





References

https://medium.com/imagescv/7-smart-techniques-for-background-removal-using-python-bb8a60fdd504#:~:text=Thresholding,the%20foreground%20from%20the%20background.

https://docs.opencv.org/4.x/d9/dde/samples_2cpp_2kmeans_8cpp-example.html

https://softwareengineering.stackexchange.com/questions/314222/how-to-implement-k-means-algorithm-on-rgb-images

https://docs.opencv.org/3.4/d5/d38/group__core__cluster.html#ga9a34dc06c6ec9460e90860f15bcd2f88

https://docs.opencv.org/3.4/d9/dde/samples_2cpp_2kmeans_8cpp-example.html

https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html

https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#ga67493776e3ad1a3df63883829375201f

https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html

https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga107a78bf7cd25dec05fb4dfc5c9e765f

https://github.com/RobuRishabh/Real_time_2D_Object_Recognition/blob/main/src/features.cpp

https://docs.opencv.org/3.4/de/d62/tutorial_bounding_rotated_ellipses.html

https://stackoverflow.com/questions/12774207/fastest-way-to-check-if-a-file-exists-using-standard-c-c11-14-17-c

https://neptune.ai/blog/knn-algorithm