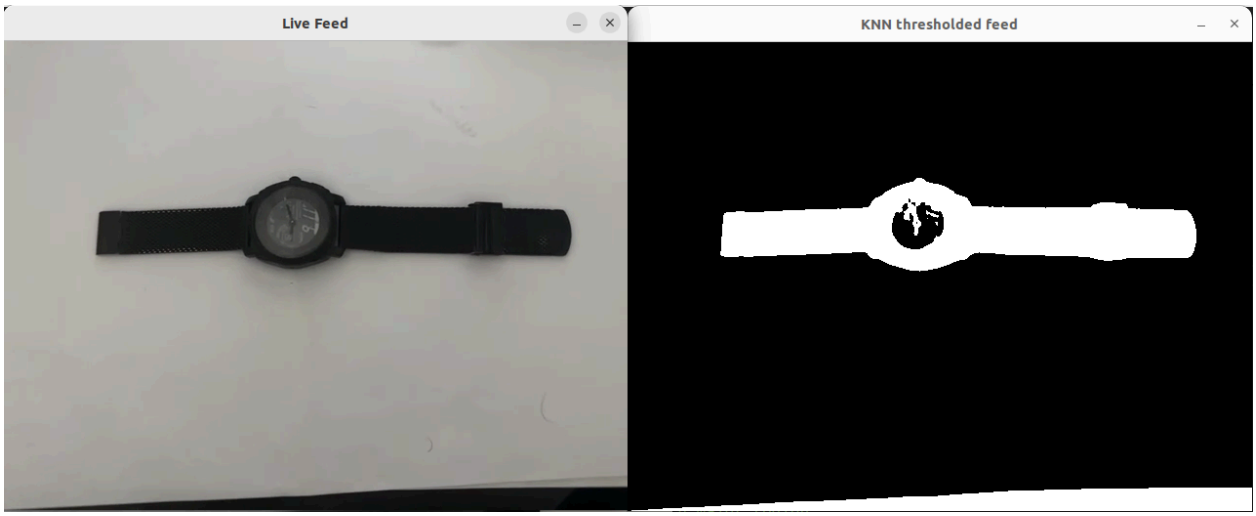
Team Members: Girish Kumar Adari, Alexander Seljuk

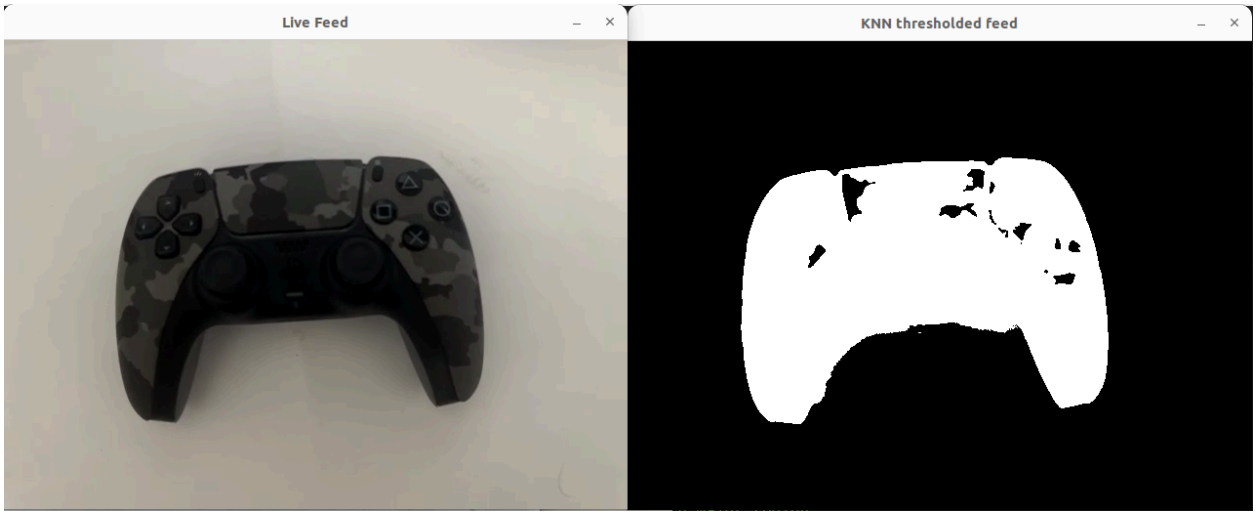
PRCV - Project 3: Real-time 2-D Object Recognition

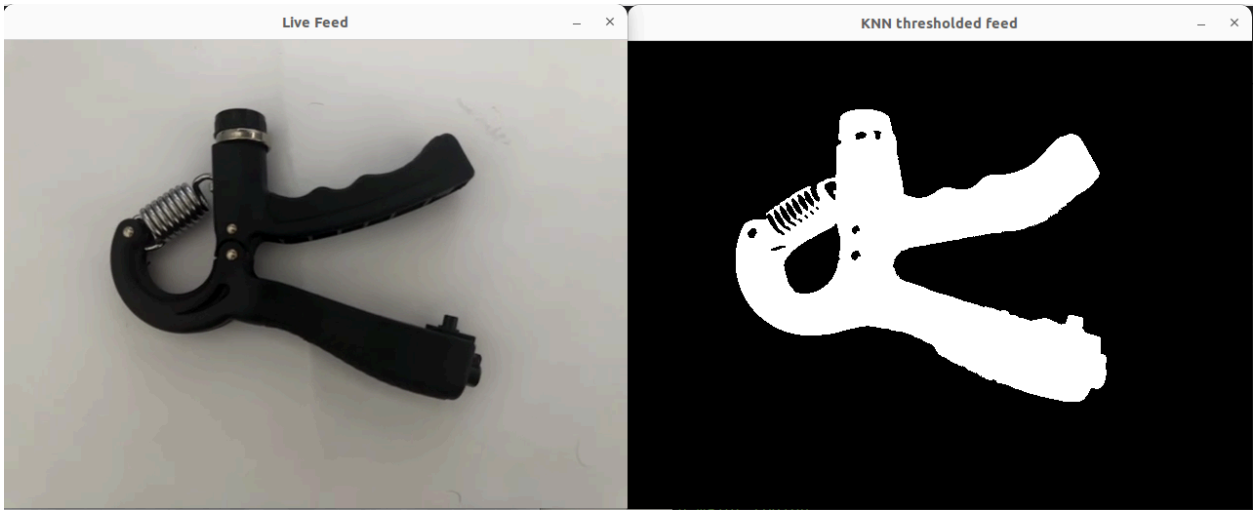
Task1: Threshold the input video

Before thresholding, the image is first preprocessed, by which it it first smoothened using custom gaussian blur method, converted to gray scale and enhanced contrast using Contrast Limited Adaptive Histogram Equalization (CLAHE) method, setting the cliplimit to 2. Implemented custom KNN thresholding technique. KNN dynamically figures out threshold and applies binary threshold based on the neighbors (inerted the image to have black as background and white as foreground).

Below are the images of original object and corresponding KNN thresholded image of the object.



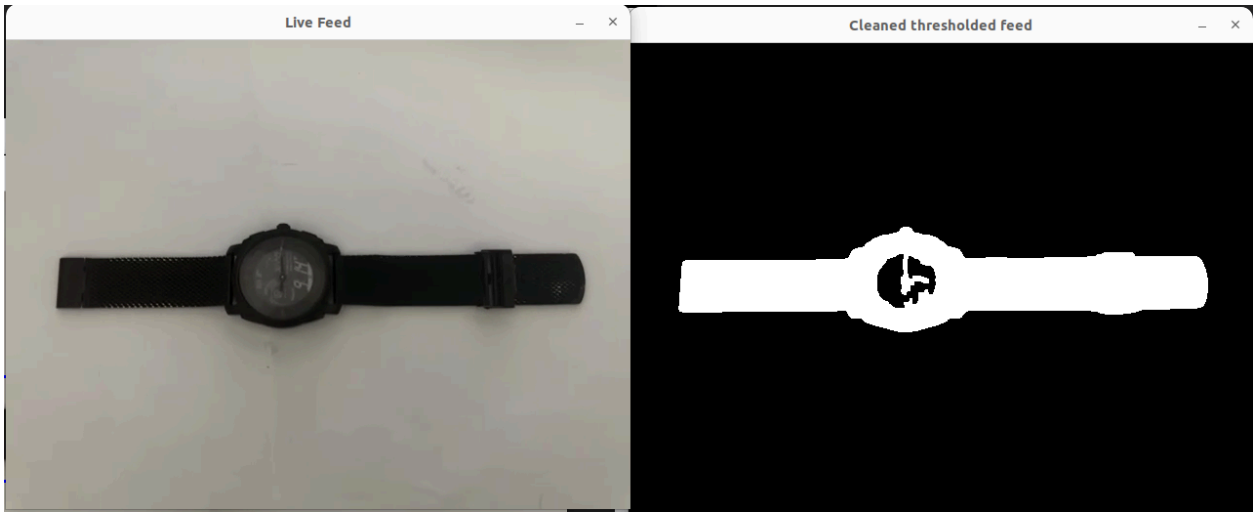


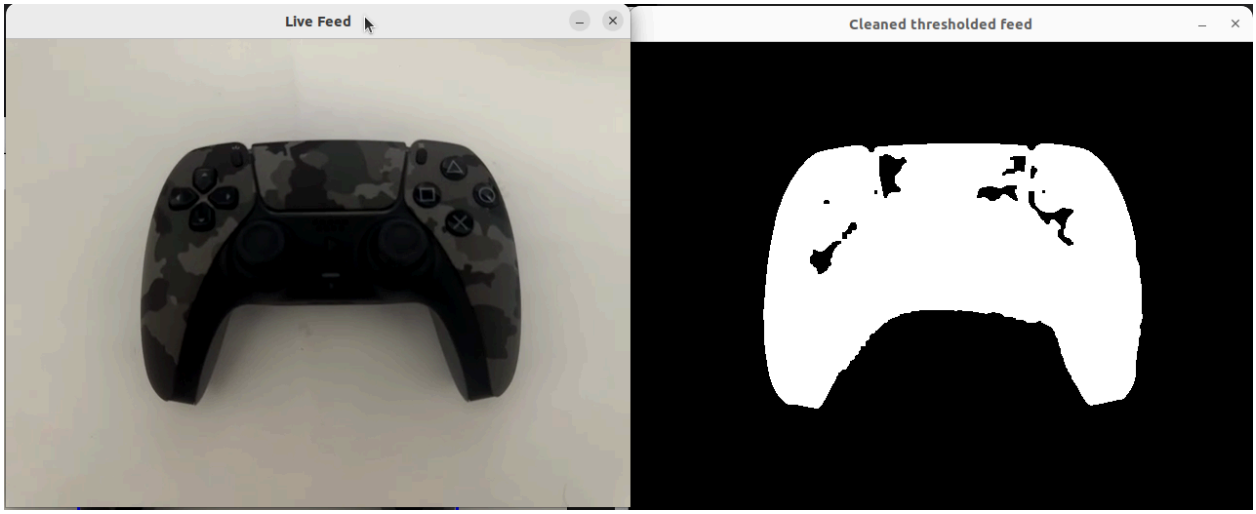


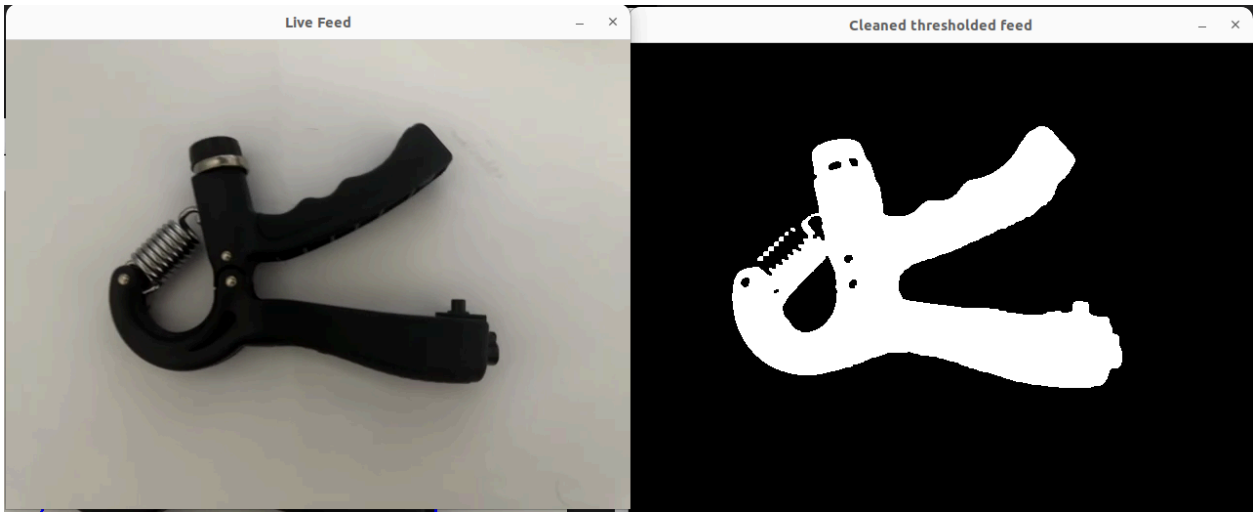
Clean up the binary image:

For cleaning the thresholded images, used 3x3 rectangular structuring element, applied opening to remove small noise and applied closing to close small holes

Below are the images of original object and corresponding cleaned KNN thresholded image of the object.





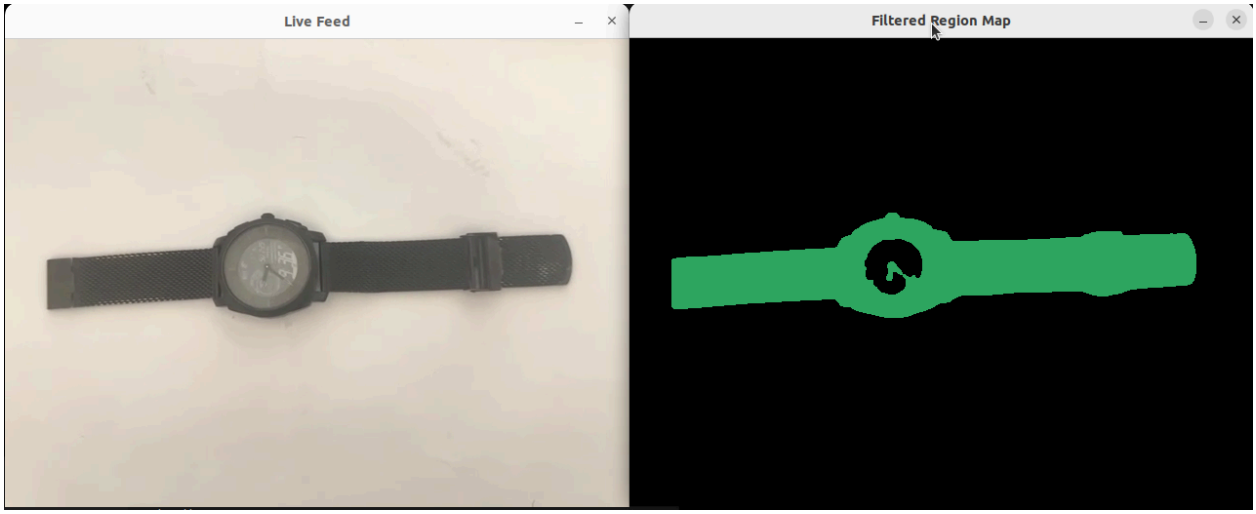


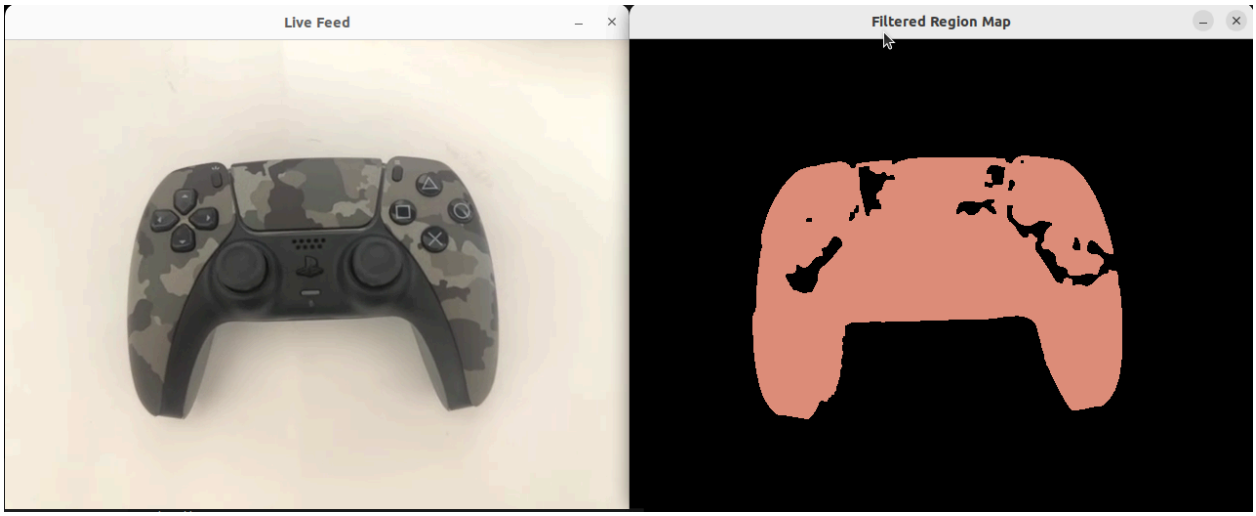
Task 3: Segment the image into regions

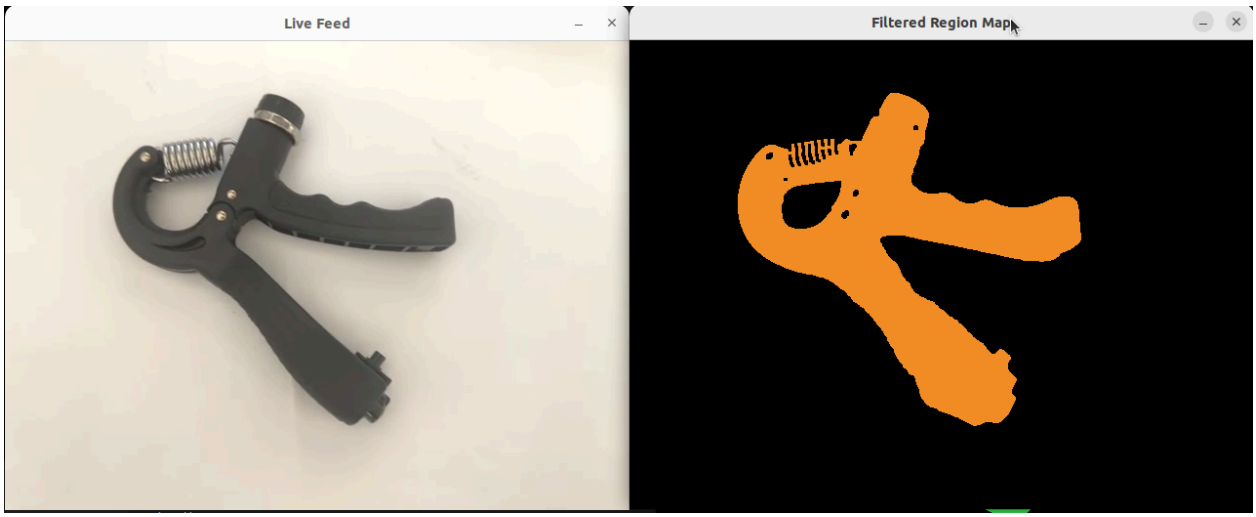
For this implemented custom regionGrowing algorithm as a method to find the regions, give them ids. Also had custom regionColor method to have a colored visulaisation of the region map generated by the regionGrowing method.

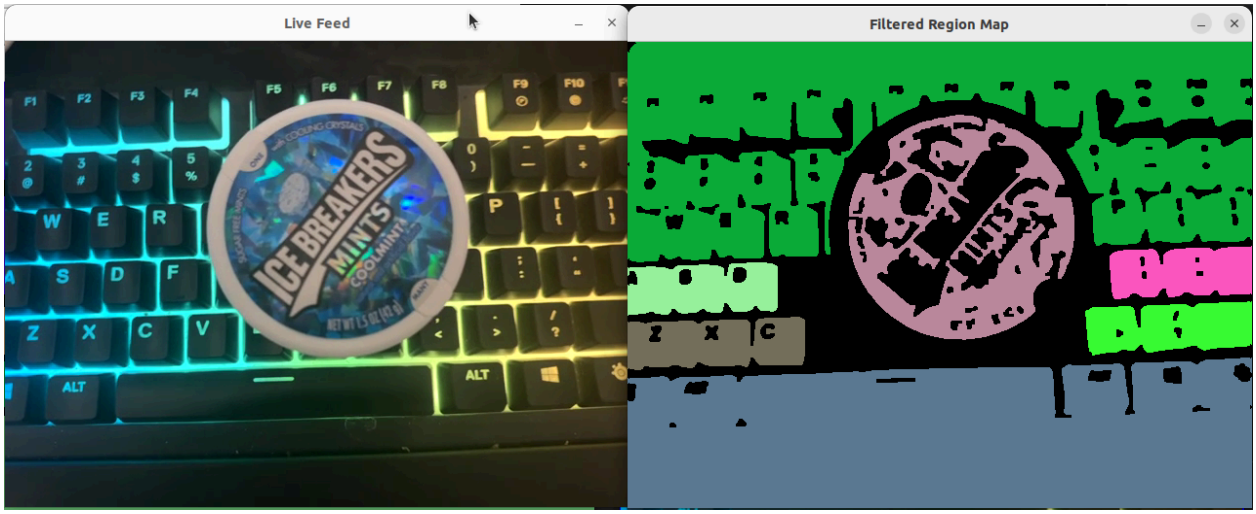
Below are the images of original object and corresponding segmented and colored region maps of the cleaned KNN thresholded image of the object.

Below are the images of original object and corresponding segmented and colored region maps after removing the smaller regions of the cleaned KNN thresholded image of the object.


Live Feed
Filtered Region Map

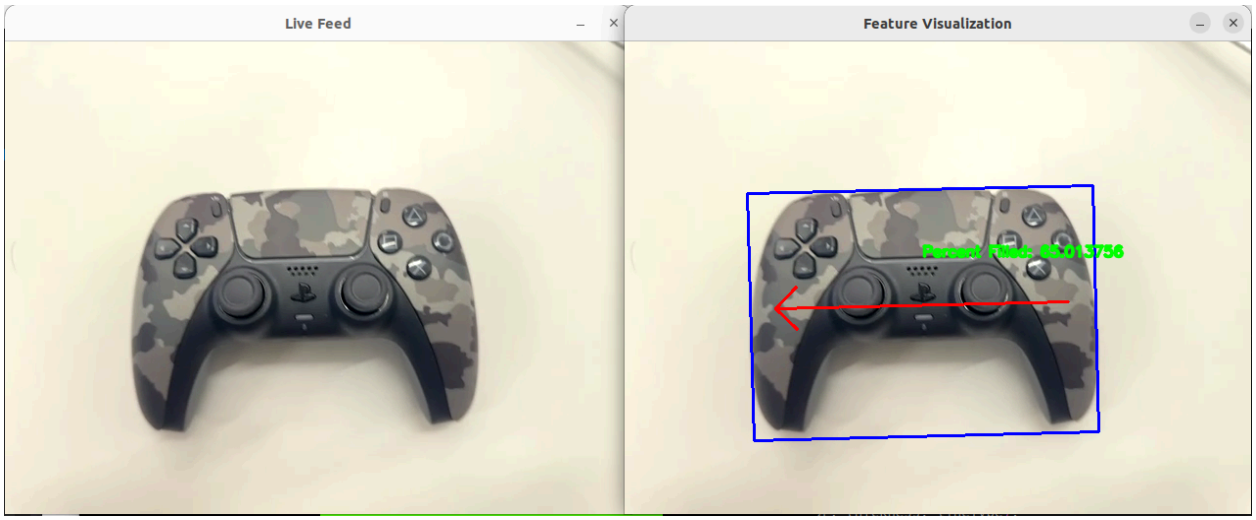

Live Feed
Filtered Region Map

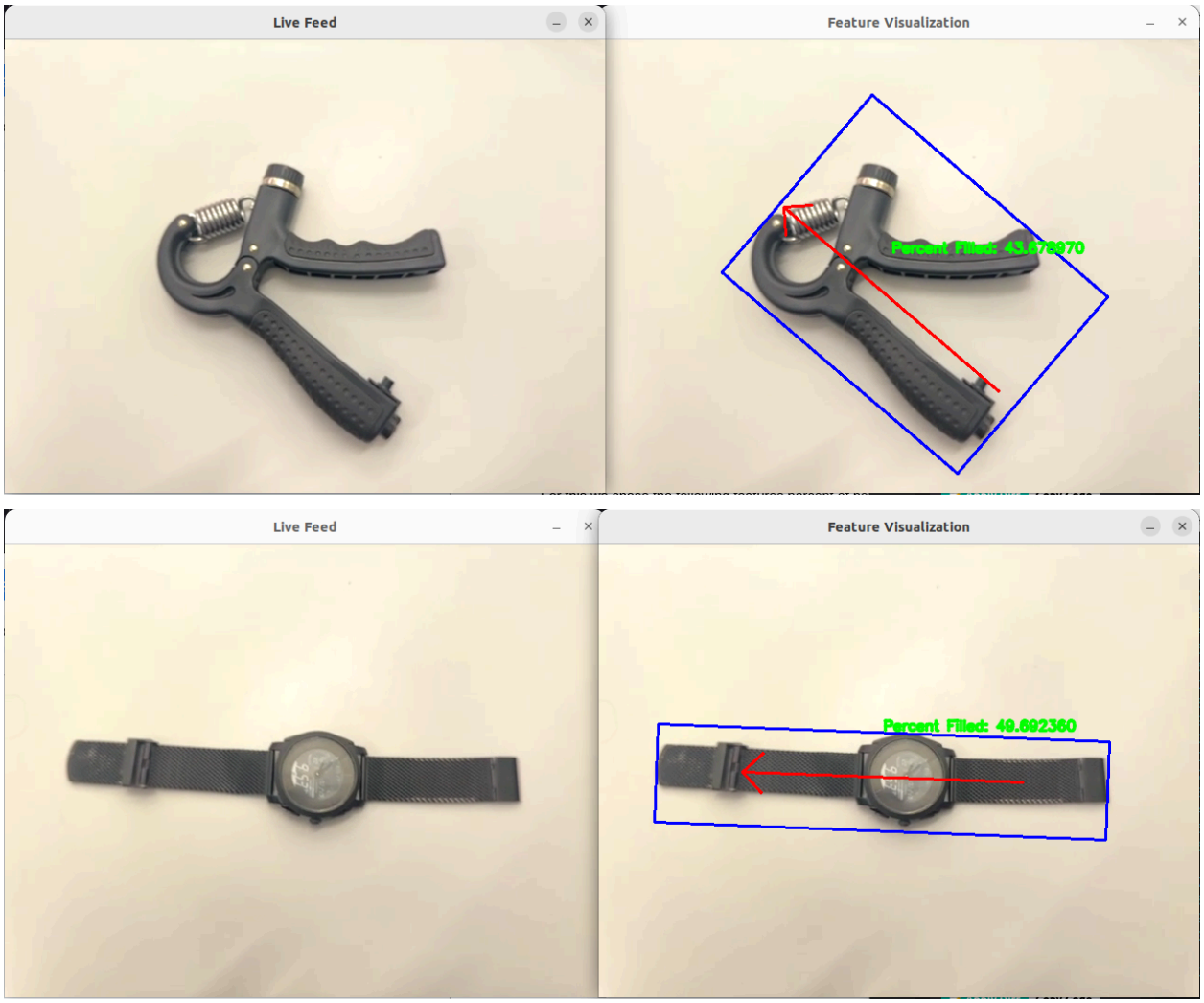

Live Feed
Filtered Region Map



Task 4: Compute features for each major region
For this we chose the following features: percent of bounding box filled, boundingBox AspectRatio, centroid of the region , main Axis Inertia, second Axis Inertia

Below are the images of the original object and corresponding detection using the bounding box and also displaying the detected regions feature, percentage filled.



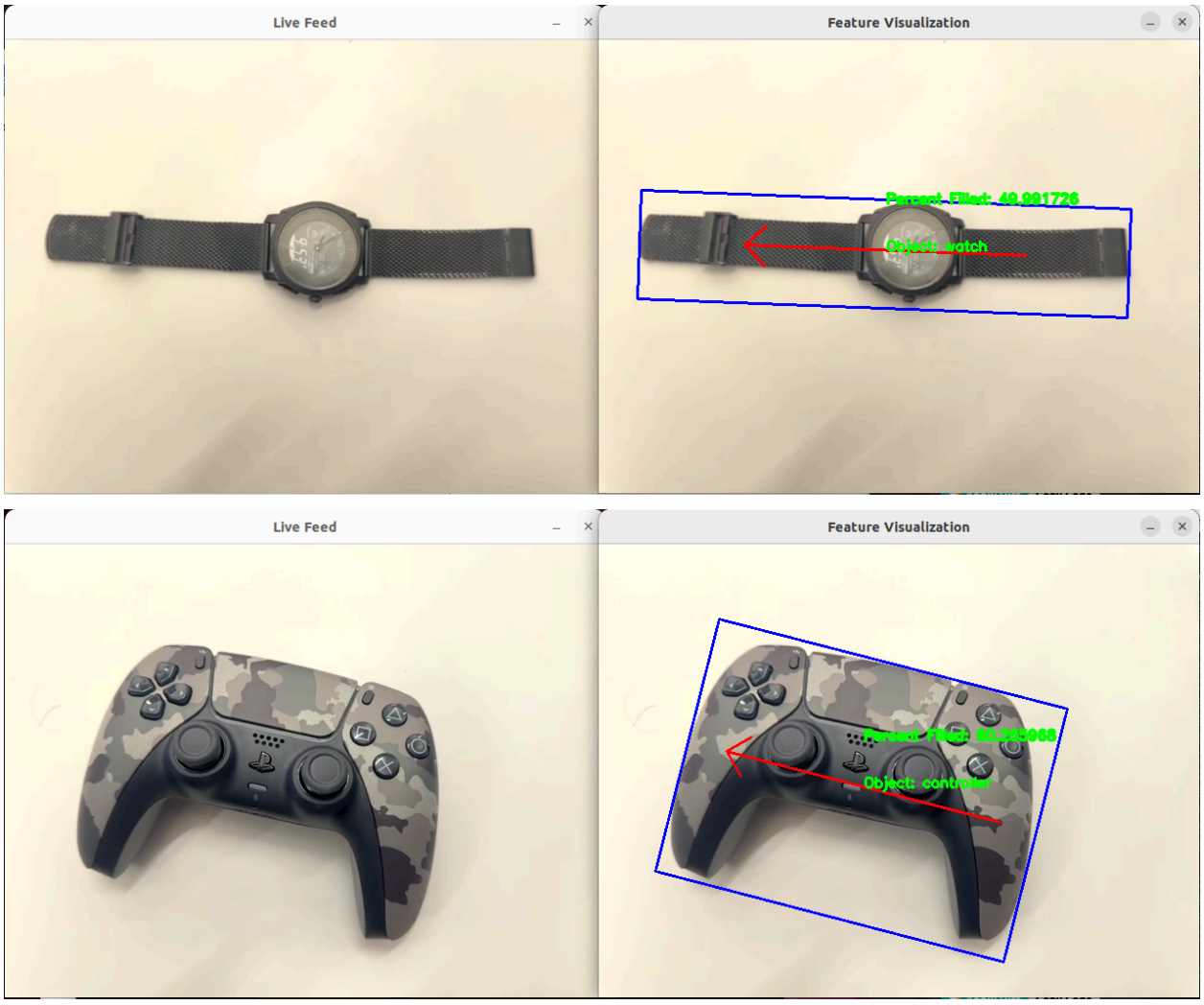


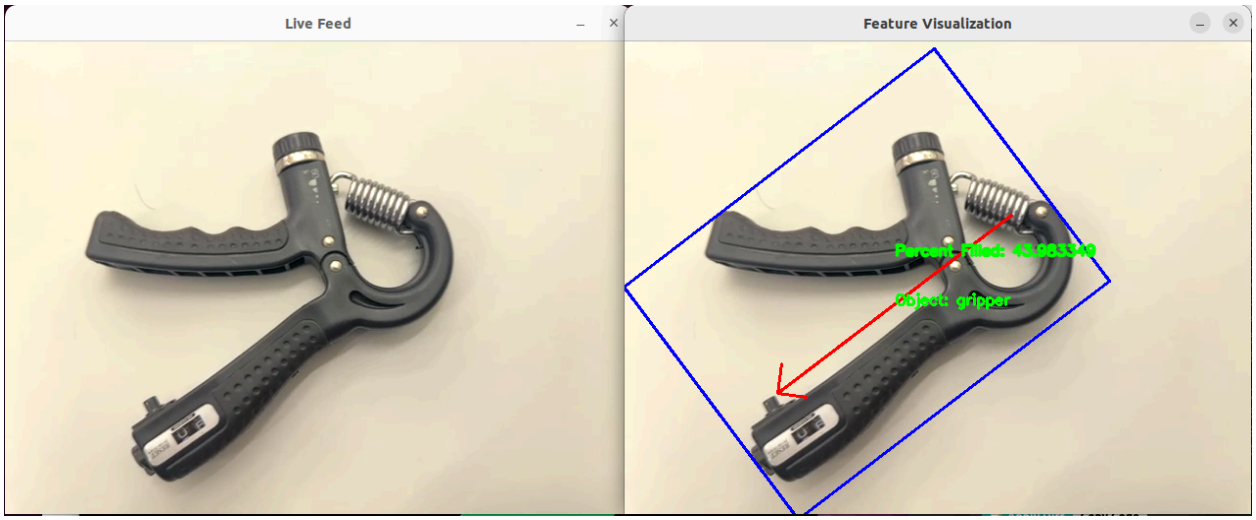
Task 5: Collect training data

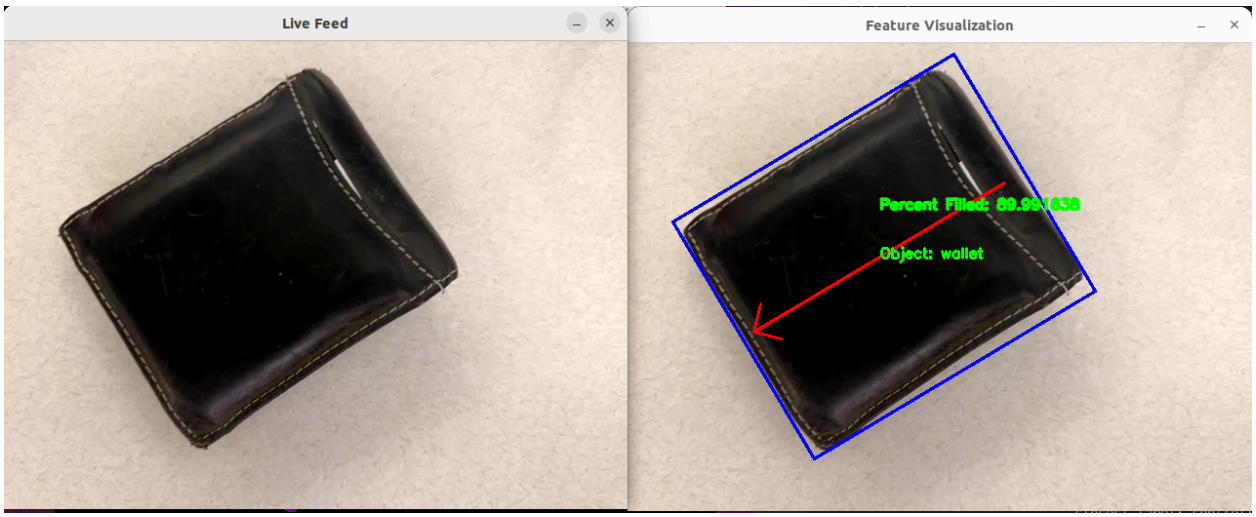
When the program is run, it's by default in default state, where tries to detect the objects without labeling them. When the d key is pressed, detect mode is activated and the program starts to classify objects, when it cant identify the object it shows unknown text in the bounding box. The user can press the key N to change the program to Training mode, the program captures the frame at which the user pressed the key, evaluates the feature vector, prompts the user to label the object. Once the user labels and the object label and its corresponding features are stored in the database.csv file and the program toggles back to default detecting state.

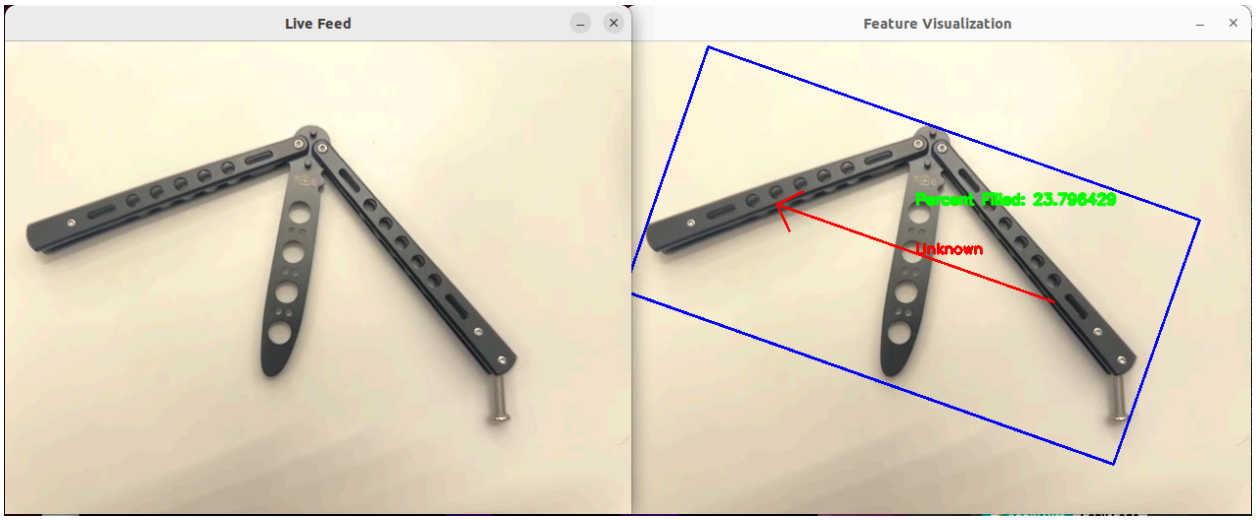
Task 6: Classify new images

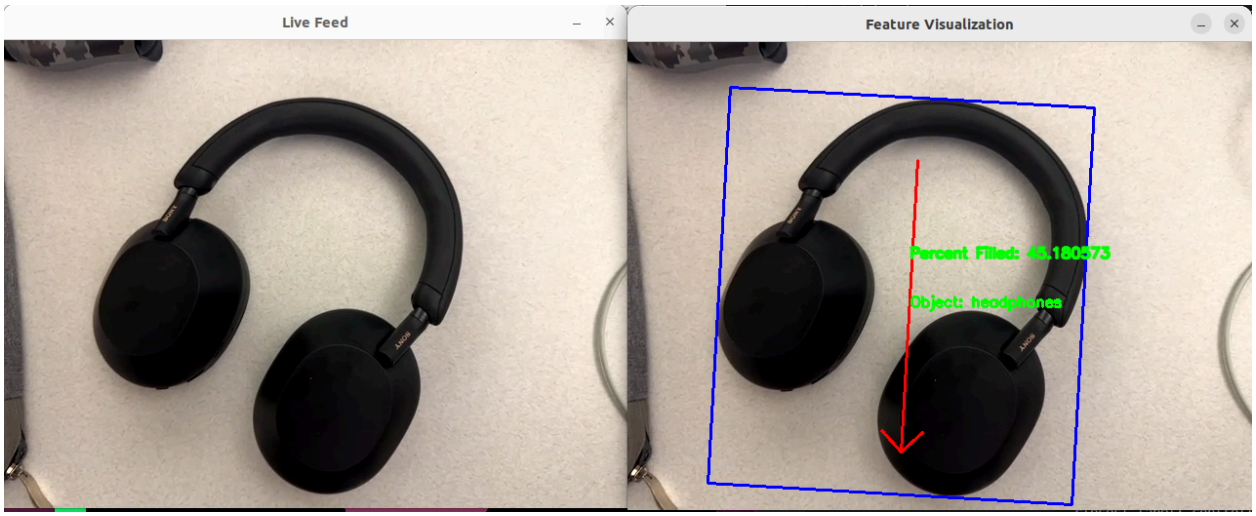
For this task, we used scaled Euclidean distance metric [(x_1 - x_2) / stdev_x] to find the closest match. We have set the threshold of minimum distance to 1.8, which gives us pretty consistent and accurate results. Below are the images of our program classifying the objects.



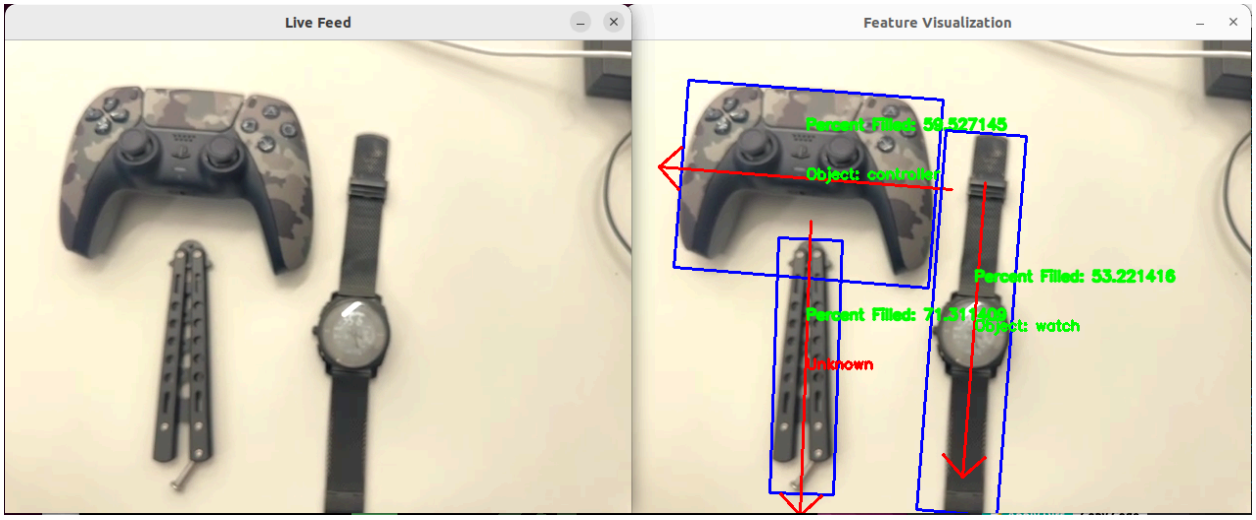

Live Feed
Feature Visualization
Percent Filled: 43.983340
Object: gripper


Live Feed
Feature Visualization
Percent Filled: 89.991638
Object: wallet


Live Feed
Feature Visualization
Percent Filled: 23.796429
Unknown



Our program can also detect multiple objects at the same time. Few examples:



Task 7: Evaluate the performance of your system

We used a confusion matrix for 5 different object each object with 3 different positions and orientations. Using the Scaled Euclidean distance metric achieved an accuracy of 93.33%.

```
Confusion Matrix:
            controller      gripper vape    wallet  watch
controller      3       0       0       0       0
gripper 0       3       0       0       0
vape    1       0       2       0       0
wallet  0       0       0       3       0
watch   0       0       0       0       3
Accuracy: 0.933333
scaled euclidean
```

Task 8: Capture a demo of your system working

Below is the link to a video demo of our application. In this demo we have shown all the different windows of displays step by step. Also demonstrated detecting, recognizing, training, calculating the confusion matrix, switching between using scaled euclidean and KNN methods of classifying the objects.

https://drive.google.com/file/d/1xASB_SqhQ9sAMcA0GJc2XOyV7elV4kti/view?usp=sharing

Task 9: Implement a second classification method

We have implemented KNN as our second classification method.

```
Confusion Matrix:
           controller      gripper vape    wallet  watch
controller      3       0       0       0       0
gripper 0       3       0       0       0
vape    1       0       2       0       0
wallet  0       0       0       3       0
watch   0       0       0       0       3
Accuracy: 0.933333
knn
```

The KNN uses scaled euclidean distance and uses k of 2. To classify we would find the k closest objects from each class and then return the label of the class with the minimum average distance.

Extensions:

1. Have implemented all the methods, algorithms (KNN thresholding, gaussian blur, segmentations, morphological operations, geometrical operators of the objects, custom oriented bounding box, bounding box features, inertia moments, scaled euclidean distance, KNN classification, confusion matrix) from scratch.
2. Our application can recognize the following objects: controller, gripper, watch, wallet, mobile, telephone, headphones, pan and vape.
3. Our system is enabled to learn new objects automatically by first detecting whether an object is known or unkown, and then collectis statistics for it if the object is unknown when the user presses the key ‘N’. We demonstrated how quickly our DB is updated and our system is able to detect the object in the video. Example of recognizing the butterfly knife.

Reflection of what we learned:

1. Learned about different thresholding methods like Binary, Otsu, KNN.
2. Learned how images can be preprocessed, segmented using regionGrowing algorithm
3. We learned how to remove the noises, which morphological operators like opening, closing are to be used.
4. We got to explore different geometric properties like inertia, orientation, centroids, box fill percentage, aspect ratios and how and which to use to classify objects and acquire the best performance.
5. We tried out different matching methods and evaluated their performance.

To sum up, this assignment enabled us to learn the basics of object classification. We acquired hands-on experience not only within opencv but also an extensive understanding of all the processes as we implemented them ourselves. By building our object recognition system we deeply learned all the steps of the classification pipeline from thresholding to matching and developed a basis for understanding and applying more advanced classification methods.

Acknowledgements:

https://docs.opencv.org/3.4/d9/df8/tutorial_root.html
https://dsp.stackexchange.com/questions/35138/region-growing-algorithm
Bruce Maxwell lectures & Notes