Project-3

-Kirti Kshirsagar
- Saikiran Juttu

Project Description:

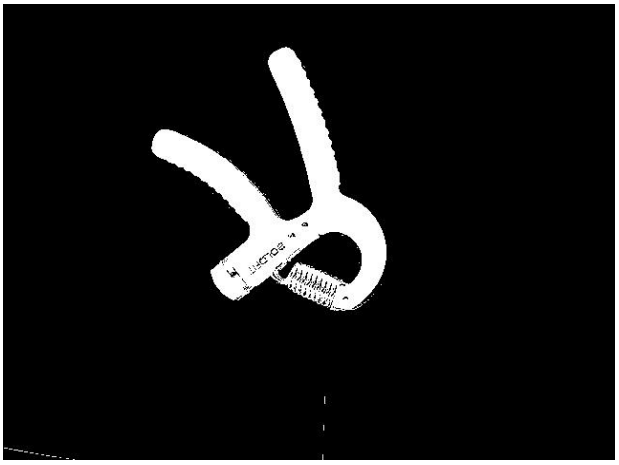
This project focuses on creating a real-time 2D object recognition system. The primary goal is for a computer to accurately identify specific objects placed on a white surface under varying conditions of translation, scale, and rotation. This is achieved by employing a camera positioned directly above the setup. The system is designed to recognize single objects within a live camera feed, distinguishing at least five different objects based on their unique 2D shapes. The process involves several stages, including thresholding to separate objects from the background, morphological filtering to clean the image, and connected components analysis to segment the image into regions. Subsequently, features are computed for each region, which are used to classify the objects using a database of known objects. The project allows for the collection of training data and evaluates the system's performance through a confusion matrix, with an extension to implement a second classification method for enhanced accuracy.

Tasks:

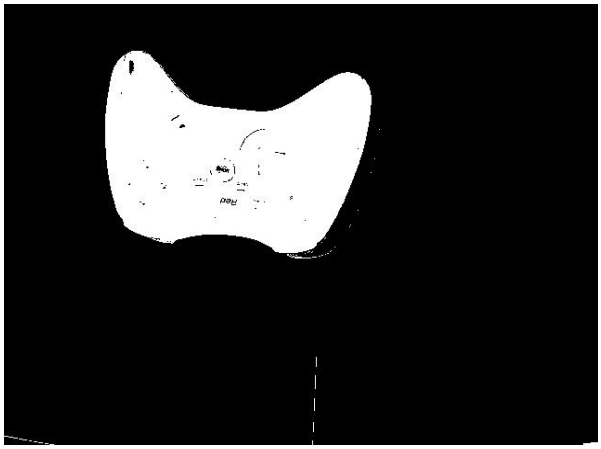
This project is a real-time web-cam based system. Our system takes a screenshot when ‘s’ is pressed and quits when ‘q’ is pressed.

Task-1: Threshold the input video [This task has been coded from scratch**]**

For the task of thresholding the input video, we implemented Dynamic Thresholding. This task involves separating objects from the background in a video feed. Dynamic threshold is calculated using a simplified K-means clustering algorithm that samples pixel values and computes two centroids, corresponding to the dark objects and the light background. The threshold is set at the mean value between these centroids. The result is a binary image where the objects are highlighted against a uniform background and we can see that the object is shown as white and the background is shown as white.

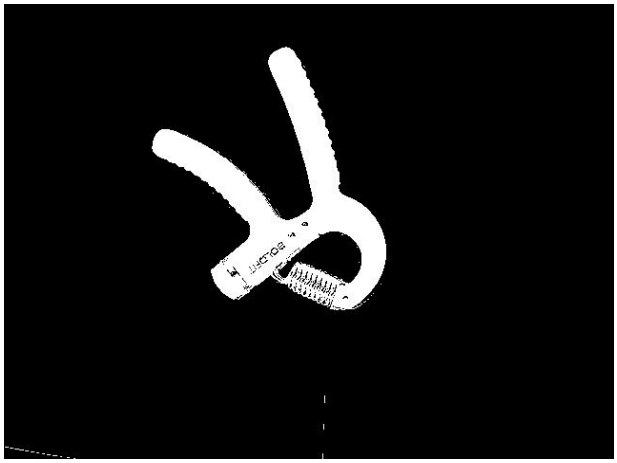


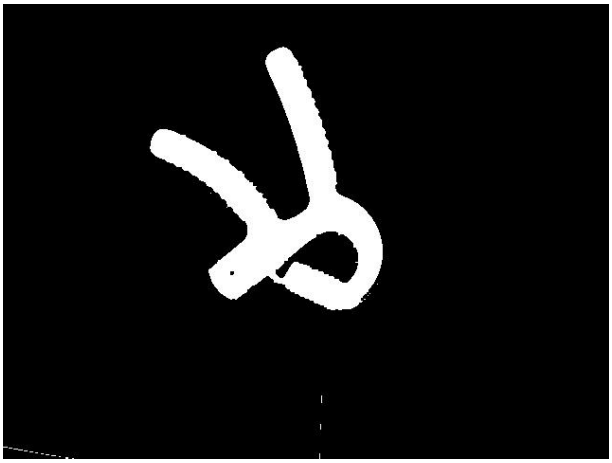
Original live feed | Thresholded Image



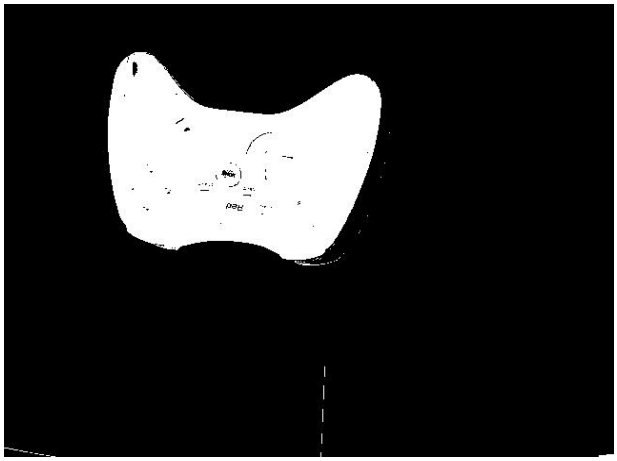
Task-2: Clean up the binary image [This task has been coded from scratch**]**

We noticed that there are some holes in our thresholded feed. To address this, dilation is first applied to fill in the gaps, followed by erosion to remove the noise and refine the object shapes. These morphological operations help in preparing the image for more accurate segmentation in the next step.

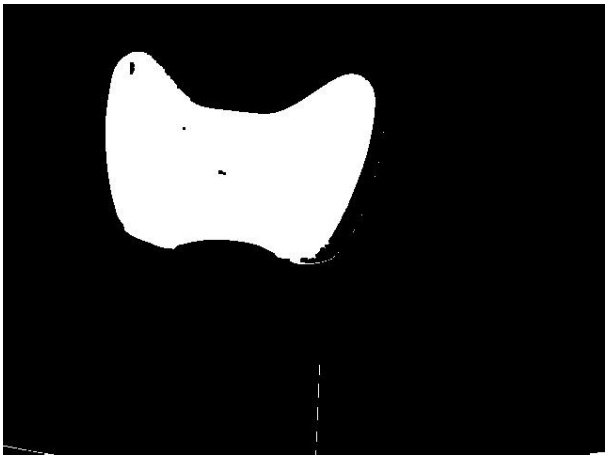




Thresholded Feed



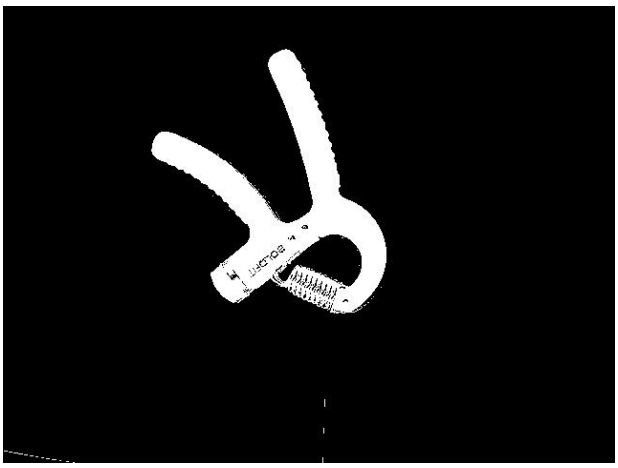
Cleaned feed



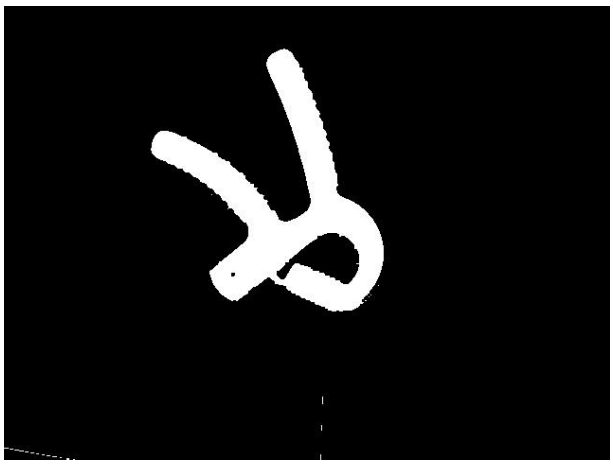
Task-3: Segment the image into regions

In this task, the cleaned binary image is processed to identify individual connected components, each representing a potential object. Using cv::connectedComponentsWithStats, each component is analyzed and filtered based on its size. Components that are too small are discarded, in order to remove potential noise. The remaining components are then colored and displayed, providing a visual segmentation of the objects in the live feed.

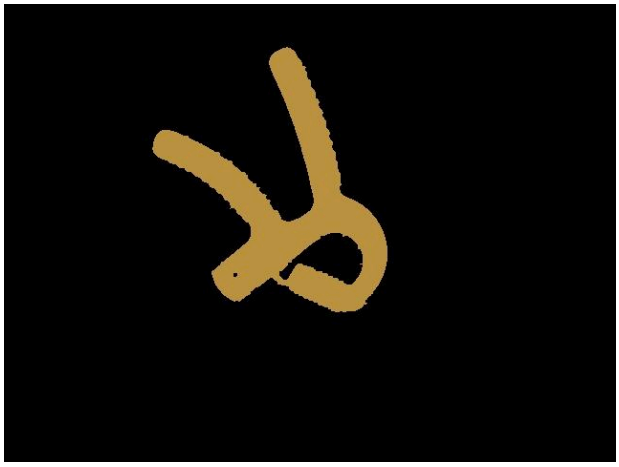
Original Feed



Thresholded Feed

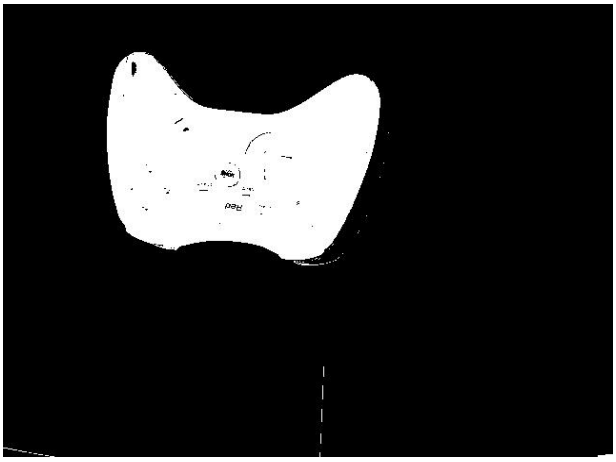


Cleaned Feed

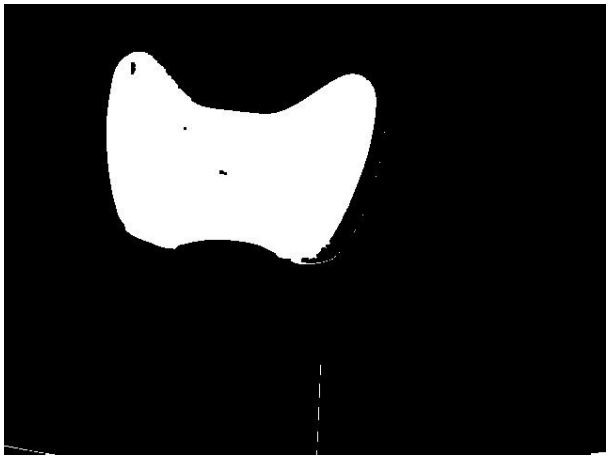


Segmentation using regions

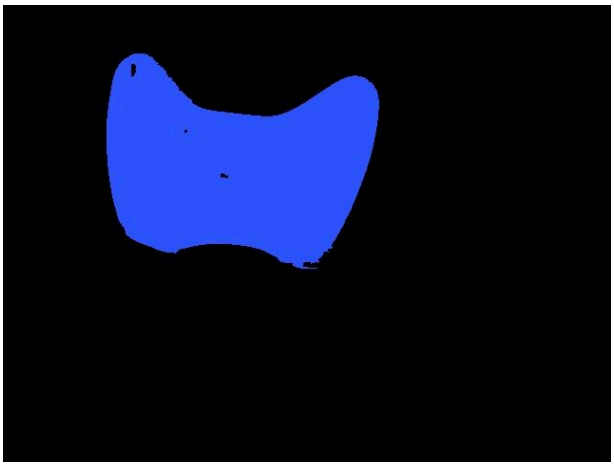
Original Feed



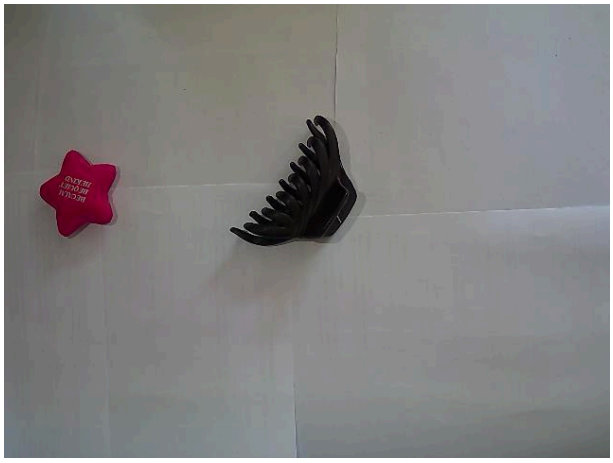
Thresholded Feed



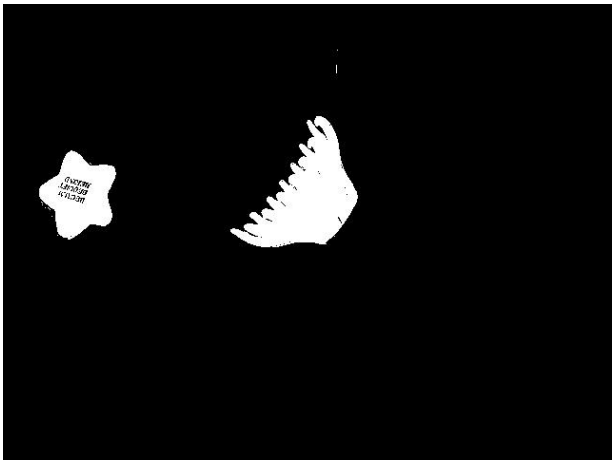
Cleaned Feed



Segmentation using regions

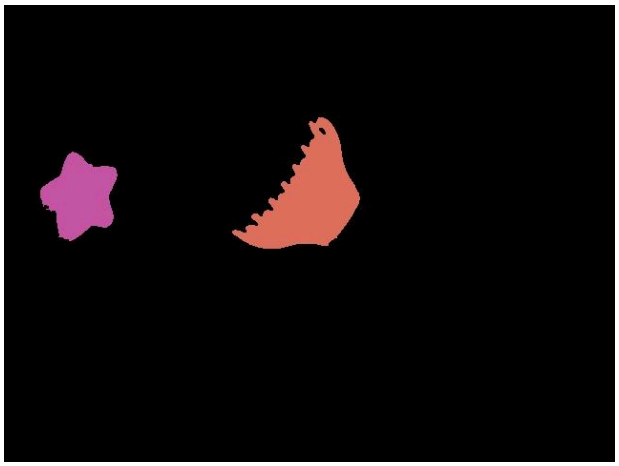


Original Feed



Thresholded Feed

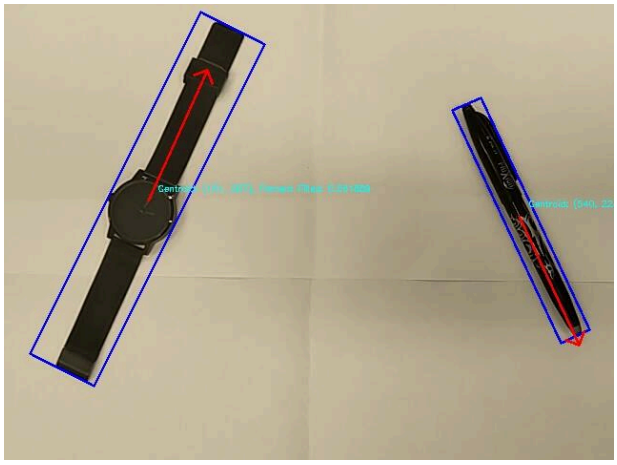
Cleaned Feed



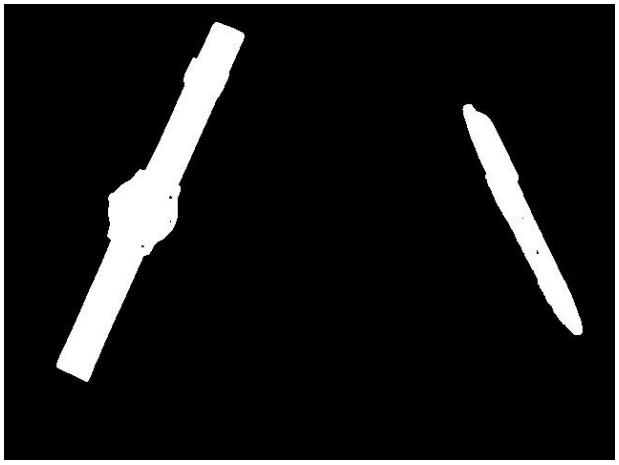
Segmentation using regions

Task-4: Compute features for each major region [This task has been coded from scratch**]**

In this task, we process the region map and region IDs to individually analyze the features of each designated region. We start by calculating moments, which are then utilized to determine the centroid of each region. Following this, we compute the central moments for every region. These calculations allow us to determine the angle of the axis of least central moment for each area. Additionally, we measure the dimensions of the bounding box (height and width) for every region and calculate the percent filled by comparing the area covered by the region to the total area of the bounding box.

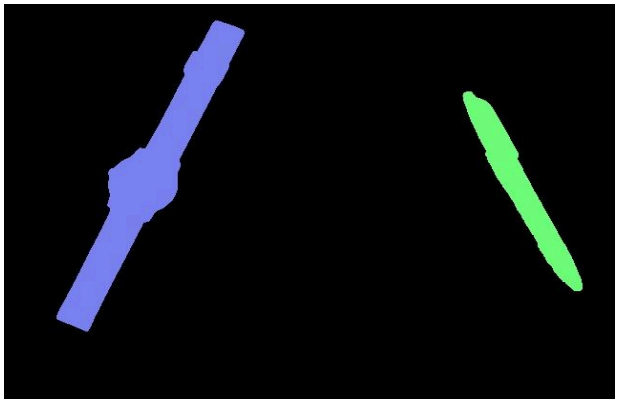


Original Feed with features



Thresholded feed

Cleaned Feed



Segmentation using regions

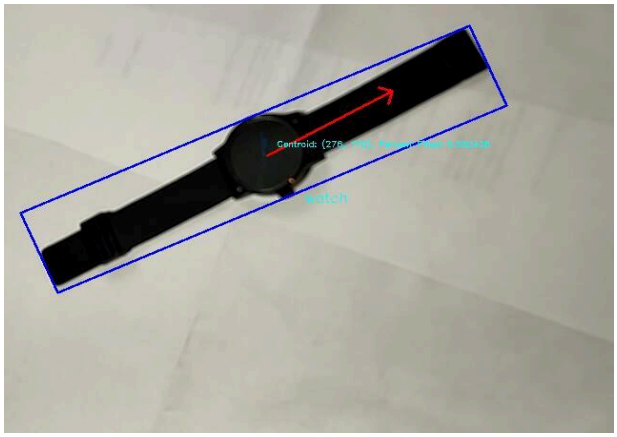
Task-5: Collect training data

In this task, the activation of the ‘n’ key in the main.cpp initiates the training mode, wherein the system captures the latest features of the object and prompts the user to assign a label to said object. A designated function is integrated to facilitate the composition of a CSV file. This function operates by initially opening the file, appending headers, and subsequently storing the labeled features within the specified CSV file.

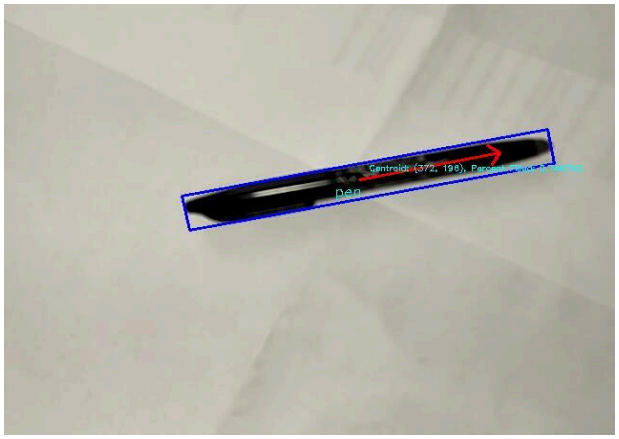
Task-6: Classify new images

In this task, triggering the ‘d’ key within the main.cpp file initiates the classifying mode. During this mode, the system captures the most recent features of the unknown object and compares them against the features of known objects. These known object features are acquired by reading the database stored in a CSV file and parsing it to compute the features of each known object. Subsequently, a scaled Euclidean distance metric is employed to compute the cumulative distance between the features of the unknown object (latest features) and those of each known object. The object with the least distance signifies the class of the unknown object.

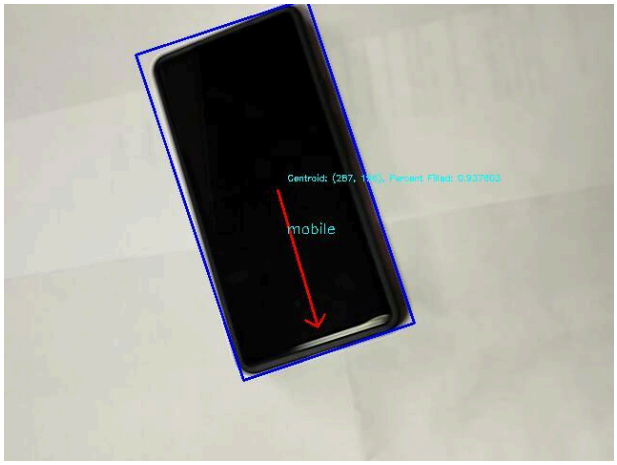
Classified Objects:



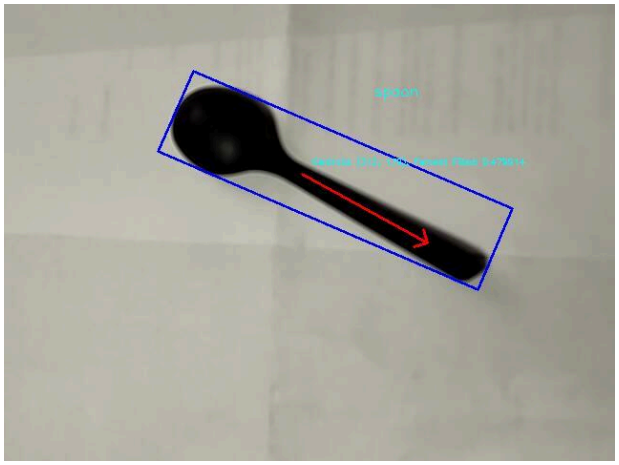
Watch



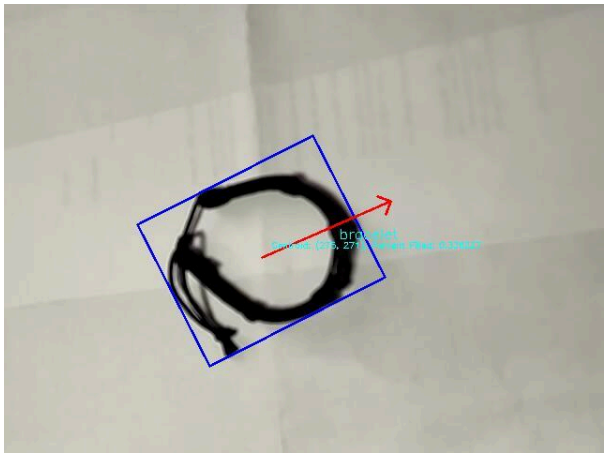
Pen



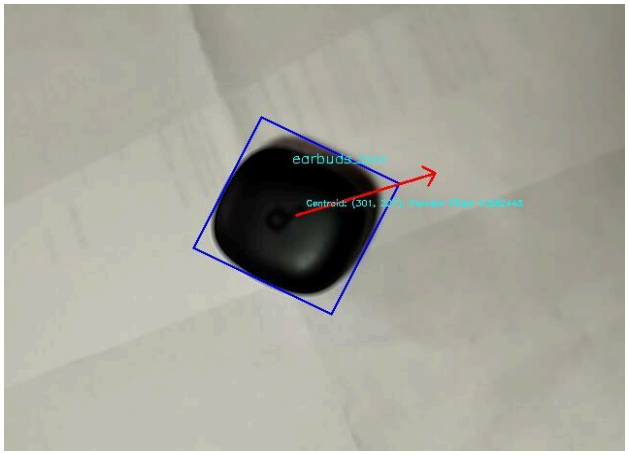
Mobile



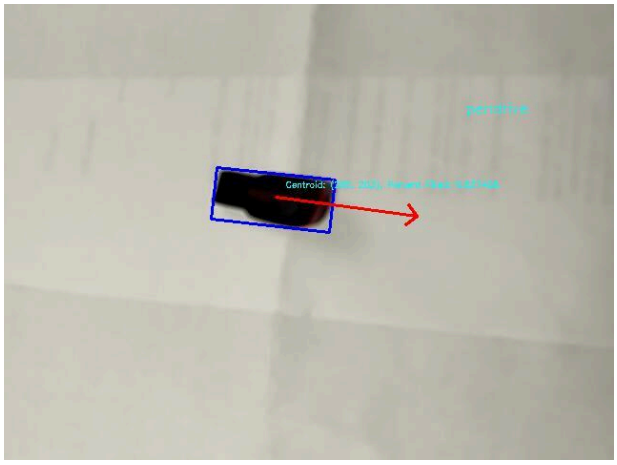
Spoon



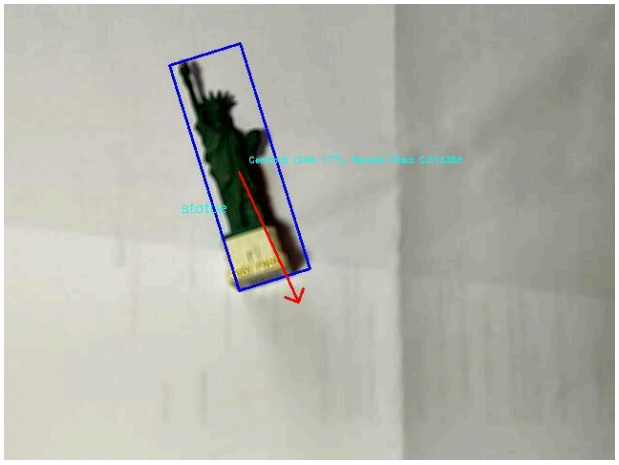
Bracelet



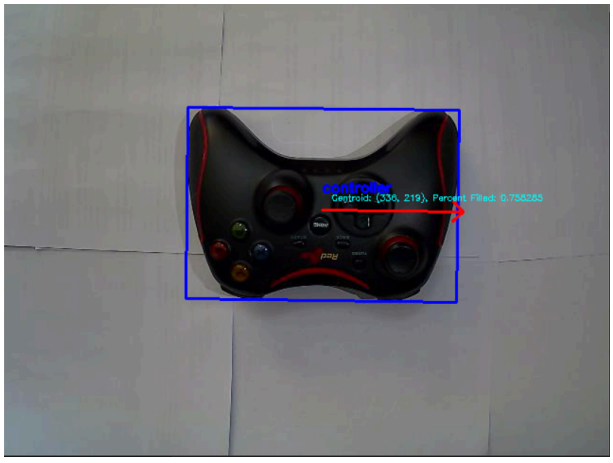
Earbuds_box



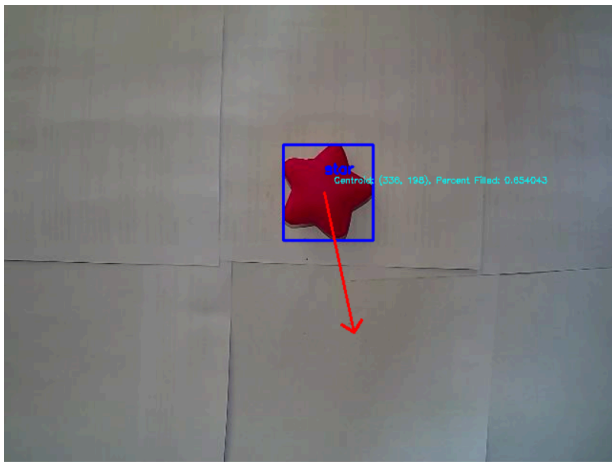
Pendrive



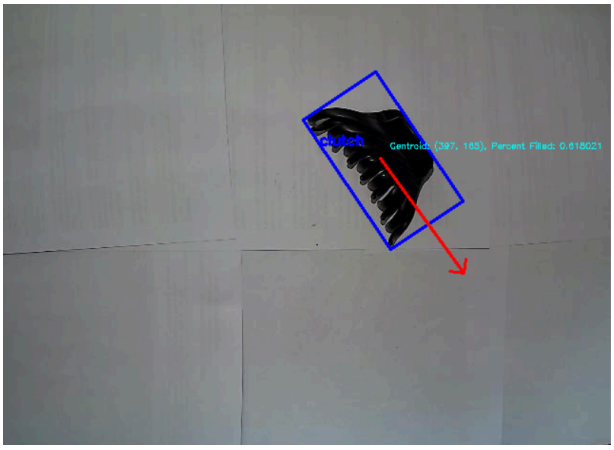
Statue



Controller



Star

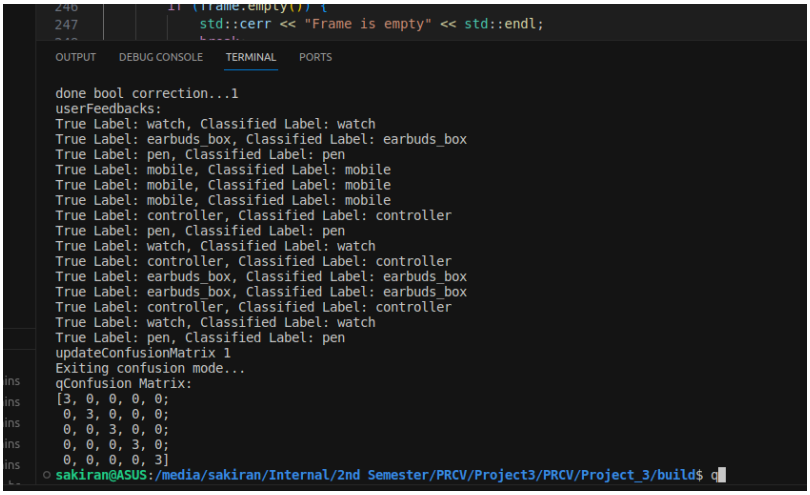


Clutch

Task-7: Evaluate the performance of your system

In this task, the activation of the ‘c’ key in the main.cpp initiates the confusion matrix calculation mode, wherein the system prompts the user to assign the true label of the object and it matches to

the classified label computed by using the distance metric in task-6. This comparison gives a bool value, which is further used to calculate the 5 x 5 confusion matrix. First this matrix is declared as zeros then it is updated by the number of correct and incorrect matches that are summarized with count values and broken down by each class.



The confusion matrix which we computed are as follows

Predicted labels

True labels					
	3	0	0	0	0
	0	3	0	0	0
	0	0	3	0	0
	0	0	0	3	0
	0	0	0	0	3

Predicted labels

True labels					
	2	0	0	0	0
	1	3	0	0	0
	0	0	3	0	0
	0	0	0	3	0
	0	0	0	0	3

No. of correct predictions =15

No. of correct predictions = 14

Total no. of objects considered = 5
In this task I took 3 different images of each object in different positions and orientations.
Total no. of predictions = 15

Classification accuracy = no. of correct predictions / total no. of predictions
Classification accuracy = 93.33 % to 100 %

Task-8: Capture a demo of your system working

This task has been implemented in the main.cpp function, starts the recording when key ‘r’ is pressed and stops when key ‘p’ is pressed. We save the demo in mp4 format.
Link to the video-
https://drive.google.com/file/d/1N5F-4br-ycPhdNtKliZ8IOUNJSyc8RH7/view?usp=sharing

Task-9: Implement a second classification method

This task asks us to implement a second classification method and then compare it to our baseline model. We considered the K-Nearest Neighbor matching method with K= 4. In this task, triggering the 'k' key within the main.cpp file initiates the classifying using KNN mode. During this mode, the system captures the most recent features of the unknown object and compares them against the features of known objects. These known object features are acquired by reading the database stored in a CSV file and parsing it to compute the features of each known object. Subsequently, a scaled Euclidean distance metric is employed to compute the cumulative distance between the features of the unknown object (latest features) and those of each known object. Objects with the least K distances are considered from these; we check for the most repeated label and display the class of the unknown object.

This method is working more efficiently than the baseline method. Because it takes the top 4 best matches and finds the class which has more matches in those top 4. In contrast, the baseline method relies solely on the single best match, potentially leading to misclassification, especially when dealing with features that exhibit marginal differences. In that case baseline matching may assign the wrong class.

Link to the video-
https://drive.google.com/file/d/1AFoqLXxugcIa1uGtxgGTe1So_fgyTtBC/view?usp=sharing

Extension:

We decided to explore many extensions this time,

1. Since we are doing the assignment in pairs, 2 of the first 4 tasks were required to be coded from scratch, we went ahead and did 3 of them from scratch, Task-1,2 and 4.
2. In the task 3, our system is to enabled recognise multiple objects simultaneously
3. Apart from the 5 objects asked in the task to be recognized, we have additional 6 objects that our system can recognize. We took around 2-3 samples for each different object in different positions and orientations. Hence a total of 31 samples are present in our DB.

trial.csv > data

```
Label,Centroid_X,Centroid_Y,Theta,Percent_Filled,BoundingBox_Ratio
spoon 387 117 -0.00410937 0.458253 3.9899
earbuds_box 296 191 -1.29074 0.870157 1.04348
controller 283 236 -0.00737873 0.729283 1.45484
watch 315 242 1.52767 0.570111 5.31373
bracelet 357 271 -0.48886 0.336563 1.18177
pen 346 234 -1.40971 0.746088 8.88172
mobile 296 261 -1.50312 0.971639 1.96365
pendrive 484 135 -1.44772 0.842193 2.42831
statue 286 166 -0.890103 0.500317 3.90186
statue 185 239 -1.0365 0.536711 3.67639
mobile 325 274 -0.415153 0.943472 2.01713
mobile 409 248 -1.37288 0.963388 1.99885
mobile 351 270 -0.0745013 0.971605 1.92623
earbuds_box 377 329 0.456383 0.867032 1.01355
earbuds_box 360 354 1.01801 0.85992 1.06164
watch 344 208 0.0168718 0.556411 5.10848
watch 293 263 0.269518 0.558098 5.04021
pen 353 329 -0.188497 0.763195 9.29352
pen 304 235 1.12508 0.774894 7.63955
bracelet 242 277 0.336843 0.394349 1.27414
bracelet 296 339 -0.95017 0.357694 1.20496
spoon 365 306 -0.0902288 0.465611 3.7082
spoon 342 296 1.20399 0.460043 3.95545
statue 309 242 0.235146 0.517353 3.44643
statue 358 259 1.47625 0.466667 2.61166
pendrive 390 233 -0.652832 0.864333 2.0636
pendrive 165 392 0.675669 0.848084 2.30496
clutch 333 171 0.0612781 0.616714 1.71113
controller 336 220 0.0193118 0.753547 1.36803
star 336 198 1.38691 0.653781 1.07447
```

Reflection:

Through this project, we learned about the complexity and challenges of real-time 2D object recognition. It was enlightening to see the importance of preprocessing techniques such as thresholding and morphological filtering in preparing images for analysis. The process of segmenting images into regions and then identifying objects based on their features highlighted the need for careful feature selection to ensure robust object recognition across varying conditions. We also got to experience how the lighting conditions affect computer vision tasks. Overall, this project offered a comprehensive learning experience in computer vision and pattern recognition, combining theoretical knowledge with practical application.

Acknowledgement of material consulted:

- OpenCV Documentation
- Stack Overflow
- Towards Data Science
- KNN matching

Installations:

So I am working on Windows 10, Visual Studio Code, Msys2, CMake, OpenCv 4 and C++.