Project-3 Report

CS 5330 – Computer Vision and Pattern Recognition

Real time 2-D Object Detection

Name: Arun Srinivasan Venkatesan Krishnamurthy, Abhinav Anil Kumar
NUID: 002839500, 002889398
Date: 02/25/2024

1. Overall project description:

The project speaks all about “Real Time 2-D Object Detection”. Our aim is to have the computer identify a set of objects with the background as a plan white surface. The set of objects when look down through a camera in the defined background should not vary when the camera is translated, zoomed in/out, or rotated on these objects. It should also be compatible when tested with a given video.

This project also requires setting an environment/workspace depending on the types of camera devices, lighting conditions and set of objects indulged in this project. Thus, setting up perfect workspace conditions does play an important role in this project.

To get this project up and running, tasks are defined. These tasks serve as essential guidelines, displaying the expected functionalities of each stage. Additionally, they act as checkpoints to validate whether the outputs align with the desired requirements for each task.

2. Task Demonstration:

1. Threshold the input video:

In this task we used video framework from our previous project. To this we framework we’ve added the K-means thresholding algorithm which separates the background from the foreground by dynamically setting a threshold value but wasn’t that useful for our project. **We coded the functionality of thresholding from scratch and the K-means from scratch**. Below images show the threshold objects from the background. In these images the objects/foreground are represented in white and background is seen as black.

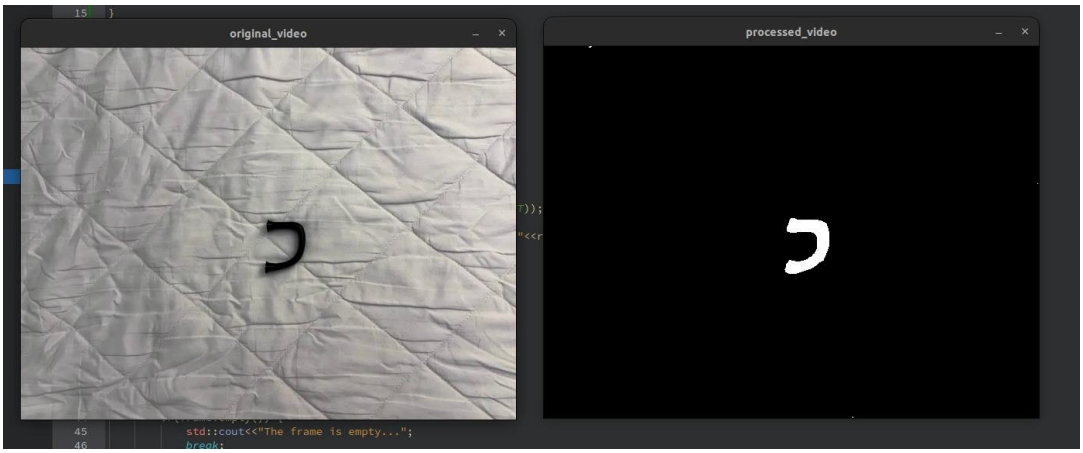


Fig.1 Thresholded “C-Shape”

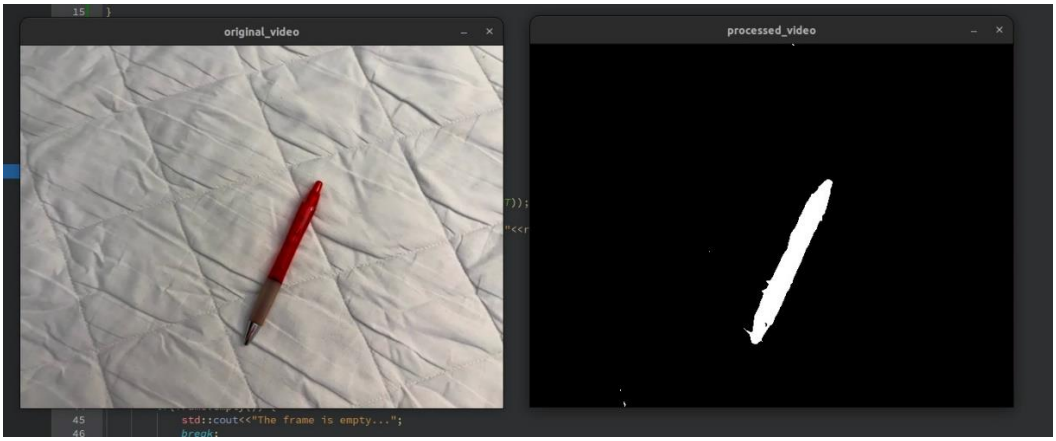


Fig.2 Thresholded “Pen” with noise

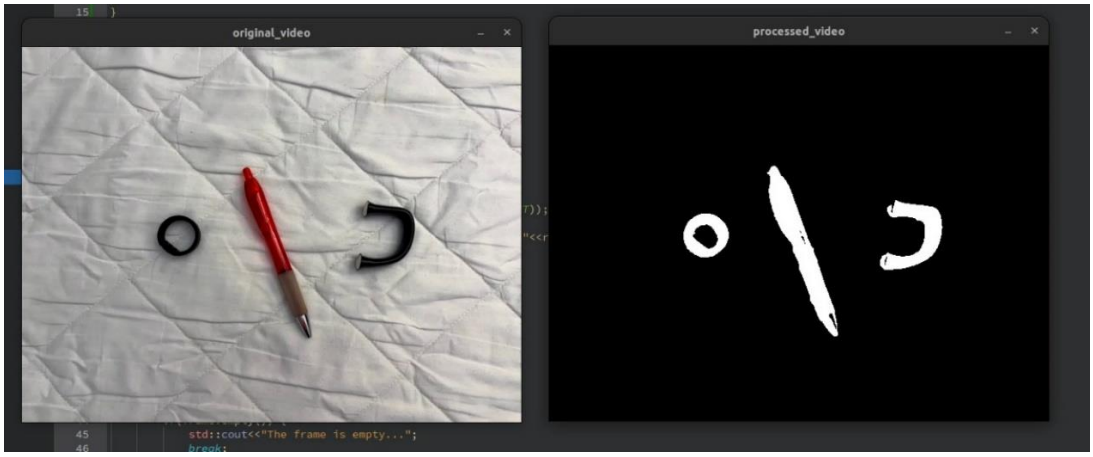


Fig.3 Three thresholded objects in one frame

Also, in this task we preprocess the image by implementing a Gaussian filter of size 5x5. This smoothens the image.

2. Clean up the binary image

In task 2 we clean up the images to get rid of any noise or small holes present in the frame. To remove these noises or small holes, we use morphological filters like erosion and dilation. Combination of erosion and dilation in a certain order can be called opening and closing operations. So, “opening” is erosion followed by dilation and “closing” is dilation followed by erosion. **We coded the functionality of erosion and dilated from scratch** and then combined it to perform opening and closing. In our project we’ve performed opening followed by closing which tends to remove small objects/noise from the foreground as seen in the images below:

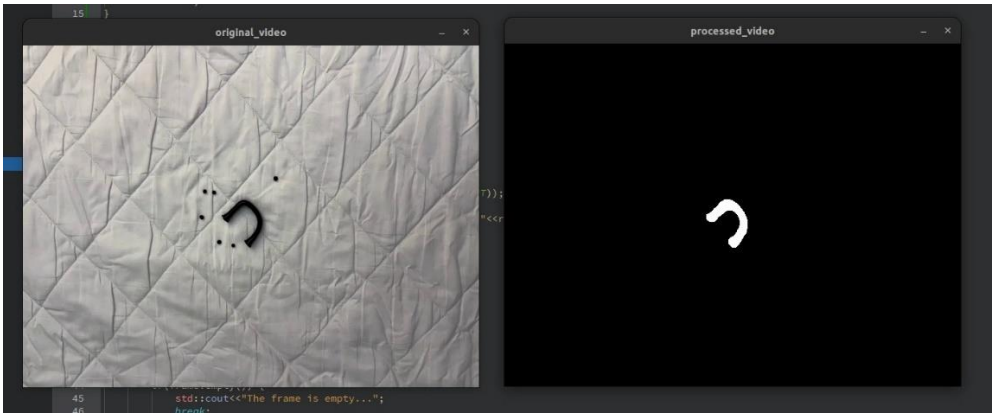


Fig.4 Morphed “C-Shape” eliminating noise in the frame.

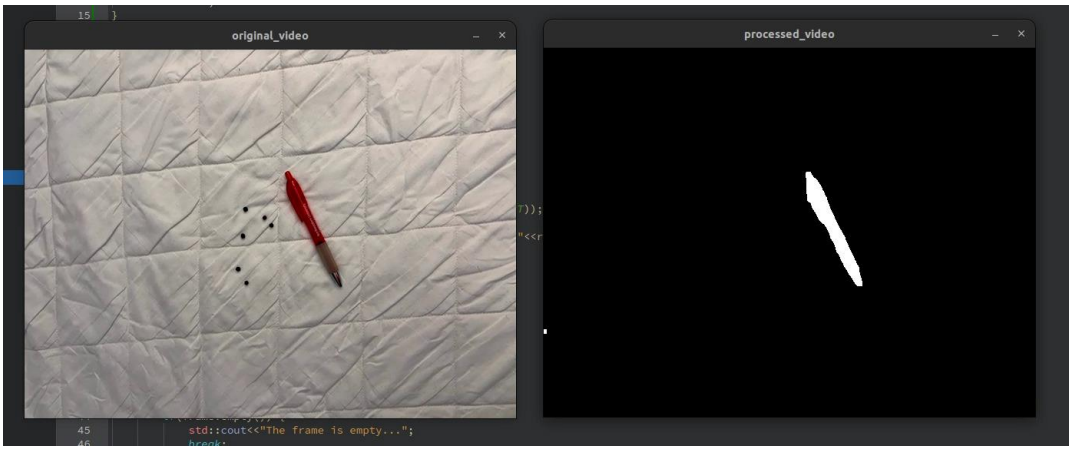


Fig.5 Morphed “Pen” eliminating noise in the frame.

In fig.4 and fig.5 we can see that video frame contains some noise which is removed and can be seen in the processed frame.

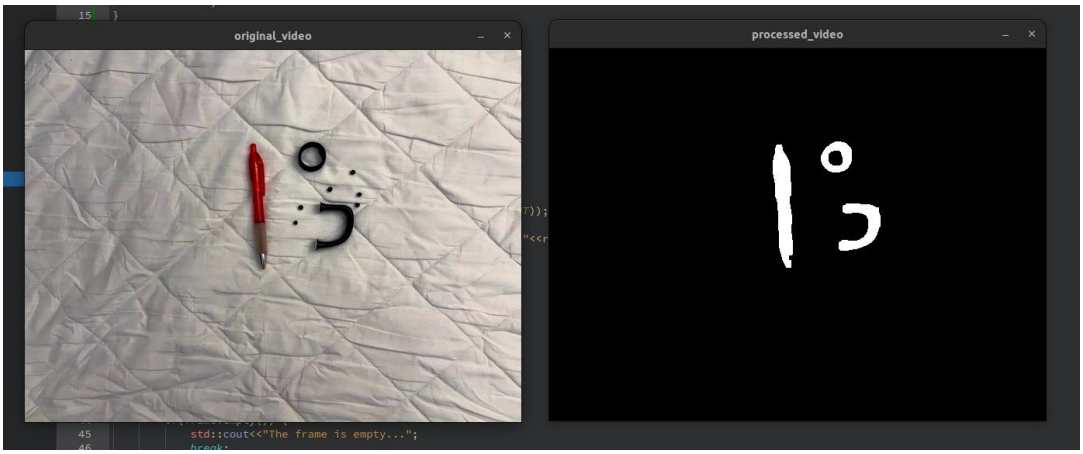


Fig.6 Morphed three objects eliminating noise in the frame

3. Segment the images into regions

In task 3, we utilized the OpenCV function " " to calculate distinct region maps for each object detected within the frame. These region maps assign a unique label or identifier to each pixel, indicating which object it belongs to. Then, we employed the "connectedComponentsWithStats" function from the OpenCV library to compute various statistics from the region maps, such as the centroid of each region and the minimum area rectangle bounding each object.

These images below show the segmentation of the object present in the frame. The objects are segmented and highlighted in blue:

Fig.7 Segmented “C-Shape”

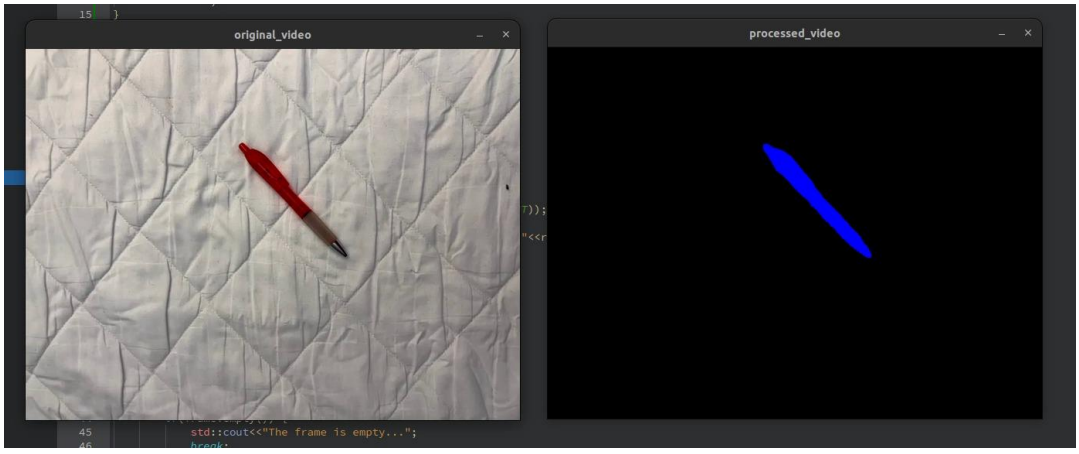


Fig.8 Segmented “Pen”

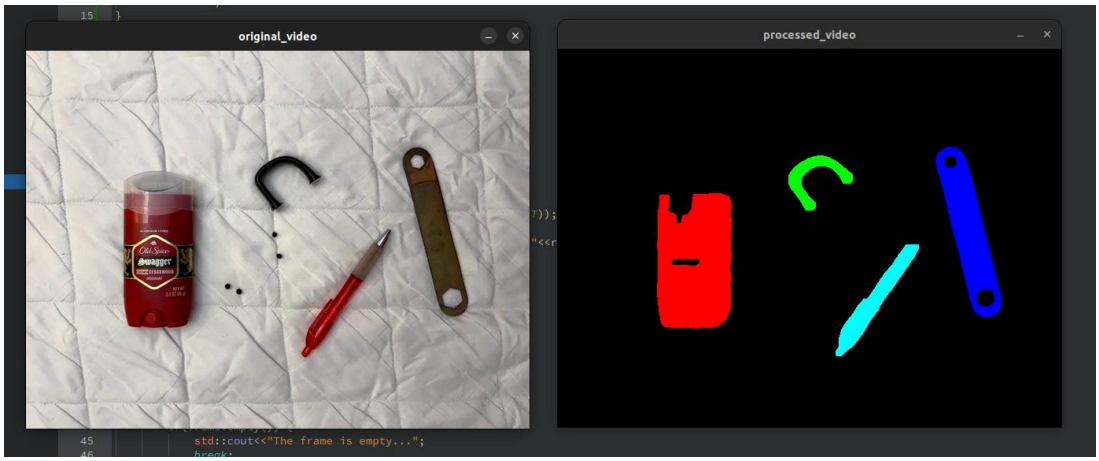


Fig.9 Multiple segmented objects

As you can see in the Fig.9 that if multiple objects are detected, then each object is labelled in a different code. We’ve set the code the assign colors for a maximum of 7 objects. If more than 7 objects are present all at once in the frame then it fails to assign colors to the excess objects present.

4. Compute features for each major region

In this task we basically compute the axis of least central moment and the orientated bounding box. The features computed in this task are “%filled and width/height ratio. We also calculated moment using OpenCV function **cv::moments**. Subsequently from this we calculated centroid location, width, height and the angle of least central moment.

Additionally, we used functions like "minarearect" and "findcontours" to further analyze and characterize the detected objects. From these functions, we derive information such as the spatial properties, contours, and geometrical features of each object, which are crucial for subsequent stages of the object detection process.

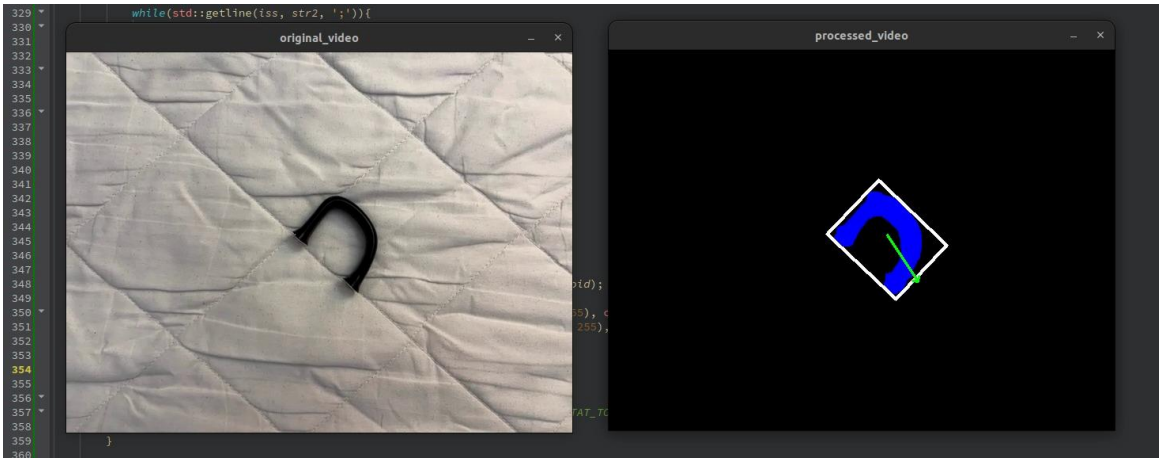


Fig.10 C-Shape with oriented bounding box and axis of least central moment

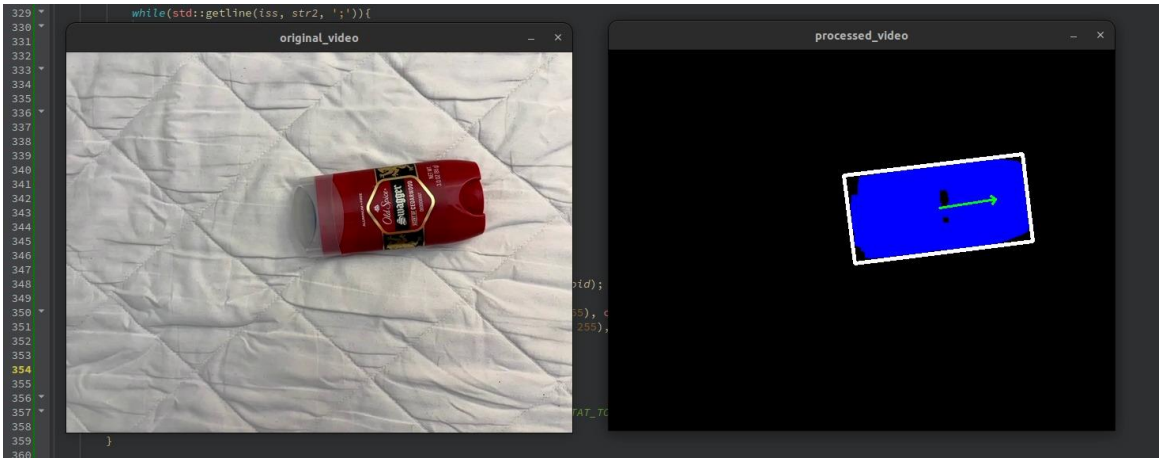


Fig.11 deo_stick with oriented bounding box and axis of least central moment

The system we developed really did a good job detecting multiple objects, segementing them, finding the axis of least central moment and finally draw oriented bounding box around all the objects. This is depicted by Fig.12 below:

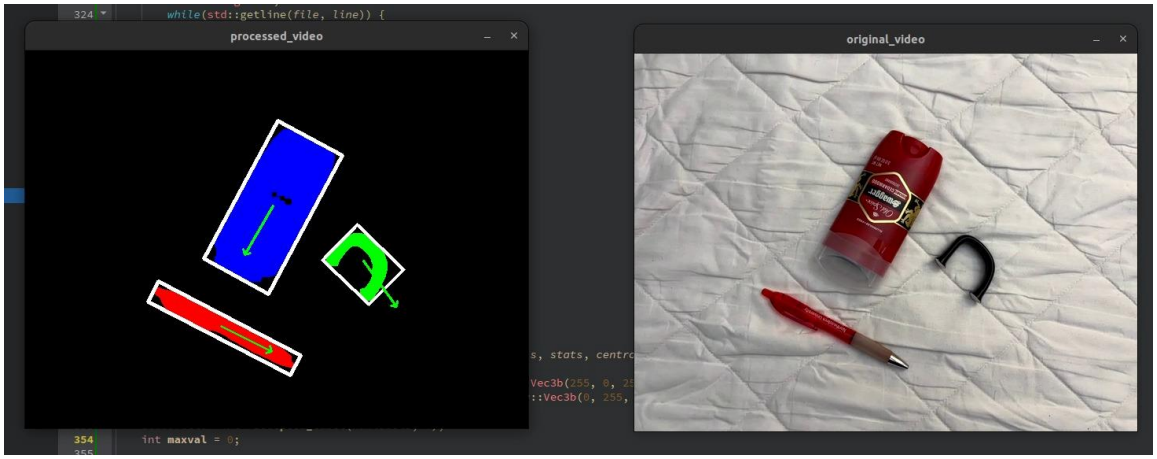


Fig.12 multiple objects in a frame with oriented bounding box and axis of least central moment

5. Collecting training data

Collecting training data set will be enabled when we are in the data collection mode. To enter the data collection mode, we've setup a key 'd'. When the key 'd' is pressed it displays the processed video window as in the task 4.

Then the user would have to press key 'n' to save the object detected by the camera. This saves the features "% filled" and "width/height ratio. We did this process n times at different orientations. The n has a max value of 5. Some of the objects used in this process required as less as 2 orientations for the object to be completely recognized. This was really a shocker for as this was happening. But this was the effect of our code and the thresholding value that we set.

With these steps mentioned above, we are all set to create a database with objects. Also most importantly, we stored the data in a text file rather than in a csv file. It was just a personal preference. The data can also be stored in csv file.

Below is an image displaying all the objects that we used to create a data base:

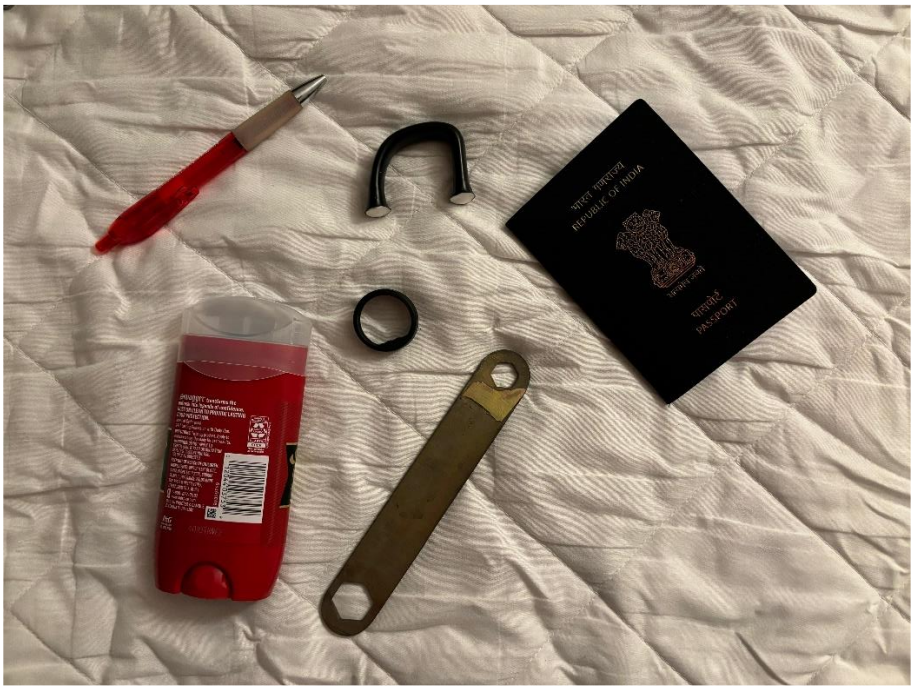
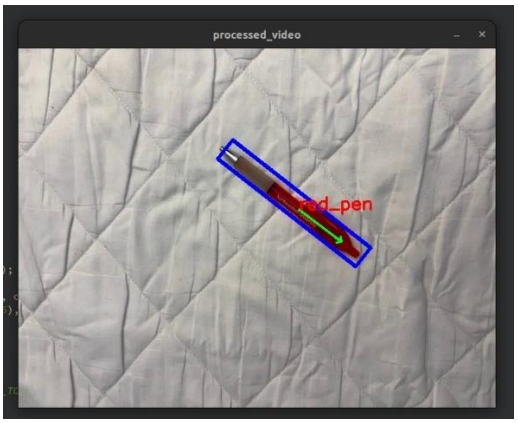
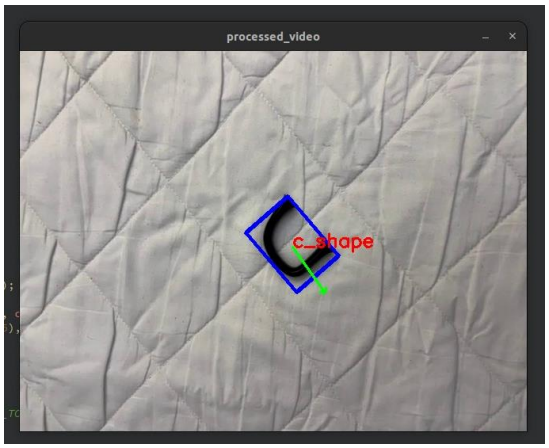


Fig.13 Image representing all the objects of dataset

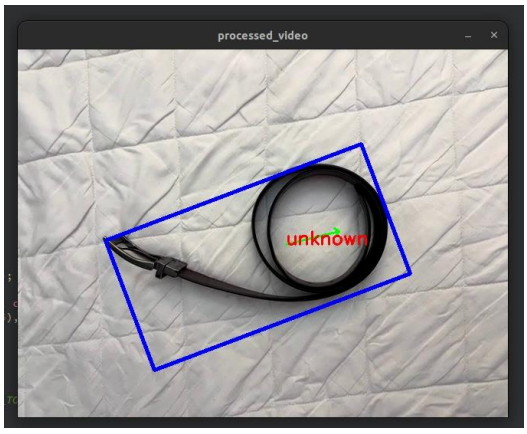
6. Classify new images

Classification of new images was done using Euclidean Distance. As we computed the features, we understood from the data that both our %filled and width/height ratio were in the range 0 – 1. This made our job lot easier as we would not have to normalize them. Images below show that all the objects are being classified with their respective labels.





Some of the observations here were unknown as the project wanted us to introduce unknown objects and wanted our system to detect if the object is known or unknown. So we've added the ability to add an object if it shows up as unkown.



7. Evaluate the performance of your system

To evaluate the performance of our system we used a 7x7 confusion matrix with the objects: C-Shape, wrench, passport, red_pen, deo_stick, plastic_ring, Unknown object

CONFUSION MATRIX: EUCLIDEAN DISTANCE CLASSIFICATION								
		ACTUAL						
		C-Shape	red_pen	deo_stick	plastic_ring	passport	wrench	glove	Unknown Object
PREDICTED	C-Shape	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	red_pen	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000
	deo_stick	0.000	0.000	0.500	0.000	0.000	0.000	0.000	0.000
	plastic_ring	0.000	0.000	0.000	0.500	0.000	0.000	0.000	0.000
	passport	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
	wrench	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	glove	0.000	0.000	0.000	0.000	0.000	0.000	0.333	0.000
	Unknown Object	0.000	0.000	0.000	0.500	0.000	0.000	0.000	1.000

From the confusion matrix above, it can be seen that C-Shape, red_pen, passport, wrench were detected as themselves or they appeared as they were predicted. But sometimes our system recognized deo_stick as passport and plastic_ring as C-Shape at different orientation, scaling, and translation. Glove appeared as passport, deo_stick, and unknown object at certain orientations which we figured to be shadows.

To avoid this we had to save the features of the deo_stick and plastic_ring at different orientations where they were recognized as some other objects. This process was repeated a couple of times. Doing this ensures that the system detects a particular object and identifies it as itself.

Some important observations and improvements after adding a couple of more data sets for the same object did the job of eliminating any ambiguity in detecting a particular object. We could see that the system did a fantastic job in this case.

8. Capture a demo of your system working

The demo video can be found in the google drive link attached below:
https://drive.google.com/file/d/15g6uBcigfUhCiSU8xwIDQws_zaKUkxFT/view?usp=sharing

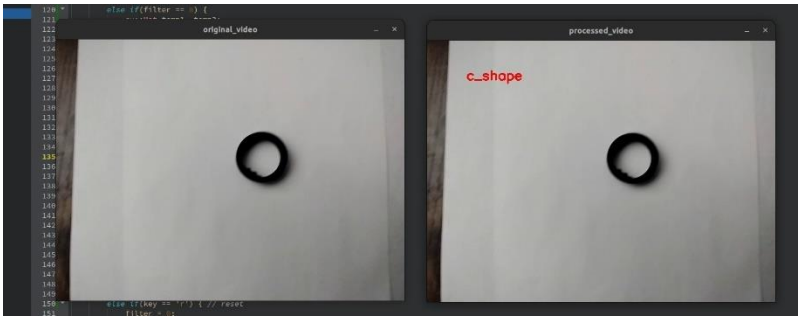
9. Implement a second classification method

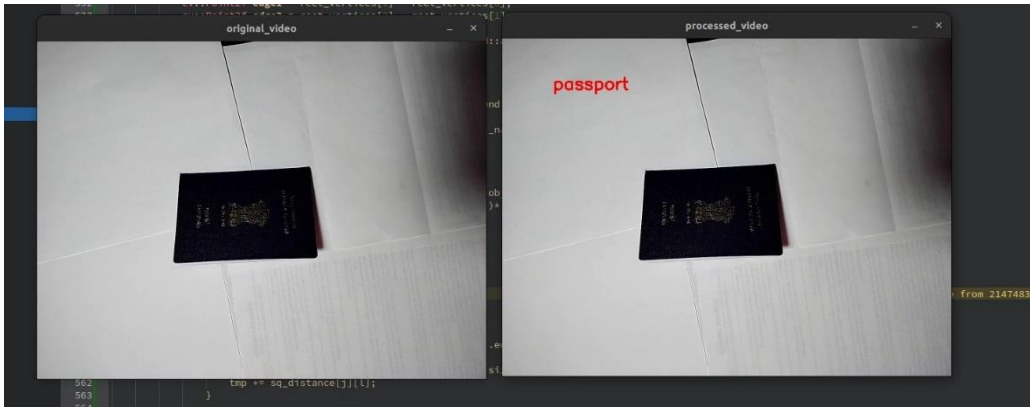
Explain your choice and how you implemented it. Also, compare the performance with your baseline system and include the comparison in your report.

We implemented kNN(k-Nearest Neighbours) as the second classification method. We chose this method as it seemed interesting. We chose the k value as 3. In this algorithm since we already have a data.txt file from the baseline system we use that file and create an array containing unique features and the algorithm uses “Euclidean square distance” to calculate the k-nearest neighbors.

We personally felt that our baseline system performed better when compared to the kNN method as it was more accurate in detecting the objects.

Below are some of the images from the kNN classification method:

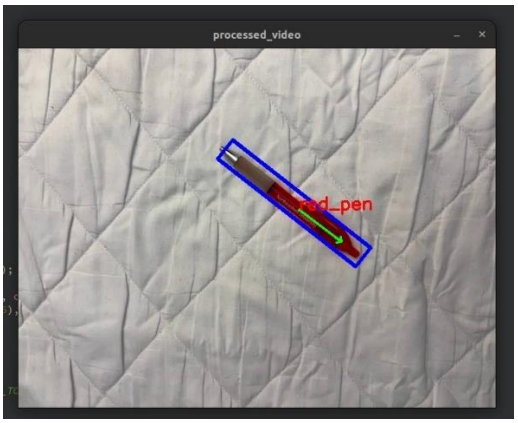


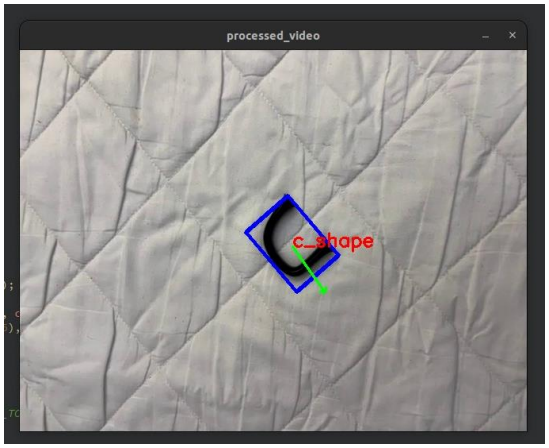


3. Extensions:

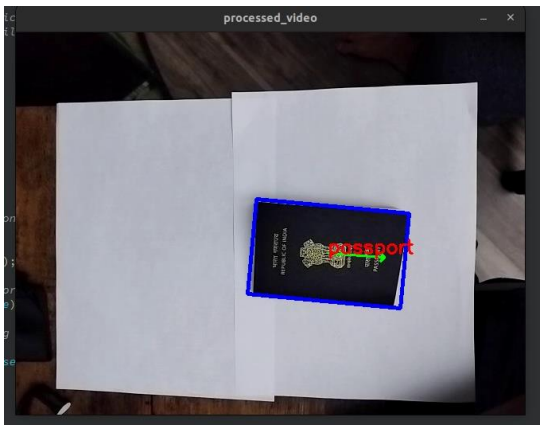
1. **Add more than the required five objects to the DB so your system can recognize more objects.**

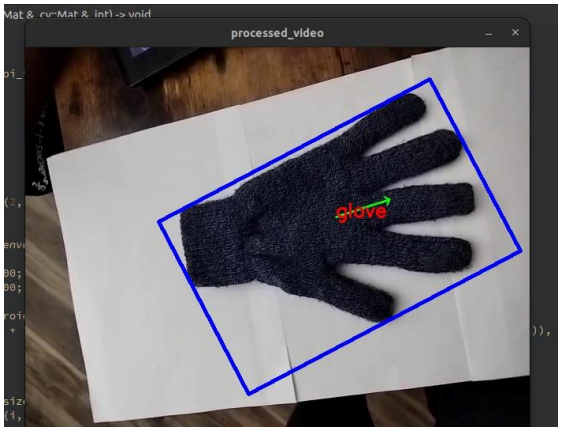
   The five objects of the database are:





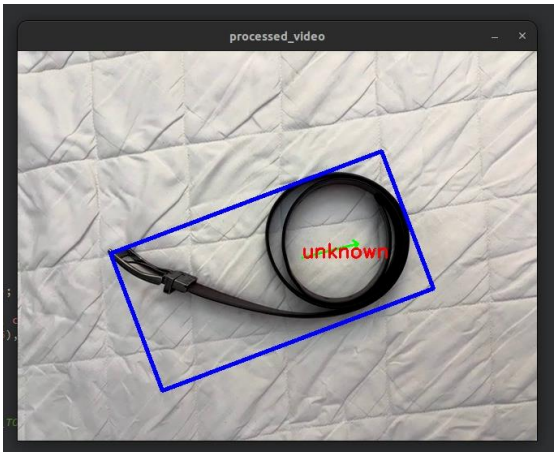
Extra items added to the database is:

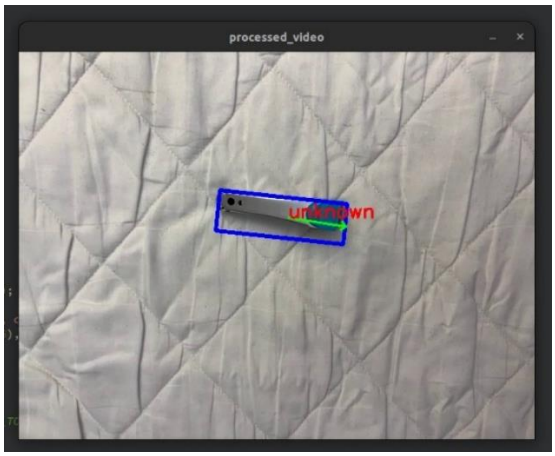




2. **Enable your system to learn new objects automatically by first detecting whether an object is unknown or unknown and then collecting statistics for it if the object is unknown. Demonstrate how you can quickly develop a DB using this feature.**

The system does a pretty good job detecting known objects from the database and unknown objects. Once the unknown objects are placed in the frame, the system can pick up the object as unknown. After this step if we were to add the unknown object, then we have a predefined set of keys that helps us add the unknown object into the database. So, once the object is placed in the frame press key 'd' to start the data collection mode and then press key 'n' it says, "Please Enter the Name of the Object". The user then inputs the object name and presses enter key to save the object. Please remember that you would have to store the data for the same object for the object to be detected. This is because whenever the orientation changes there is change in the %filled and width/height ratio as shadows are introduced.





3. **Two or more tasks implemented from scratch when working as pair**

We have implemented these tasks from the scratch:

Task1: Threshold the input video

Task2: Clean up the binary image

Task9: Implemented kNN (k nearest neighbors) from scratch

4. Key bindings to operate through the program

```
if(key == 'q') { // to quit the video ( q = quit)
        break;
      }
      else if(key == 't') { // threshold - part of task_1 ( t = threshold)
        filter = 1;
      }
      else if(key == 'r') { // reset (r = reset)
        filter = 0;
      }
      else if(key == 'k') { // threshold using k-means - part of task_1 (k = k-means threshold)
        filter = 2;
      }
      else if(key == 'm') { // clean up the binary image using morphological filters - task_2 (m = morph features)
        filter = 3;
      }
      else if(key == 'g') { // Segment the image into regions - task_3 ( g = segmenting)
        filter = 4;
      }
      else if(key == 'd') { // Compute features for each major region - task_4(d = train and capture)
        filter = 5;
      }
      else if(key == 'n' and filter == 5) { // for save a unknown object, training mode - task_5(n = save object)
        filter = 6;
      }
      else if(key == 'e') { // testing mode  - task_6
        filter = 7;
      }
      else if(key == 'b') { // knn - task_9_A
        filter = 8;
      }
```

4. Reflection:

Commencing this project, we initially grasped the fundamentals of algorithms aimed at distinguishing between background and foreground/object, such as simple thresholding and K-means thresholding. Additionally, we delved into the realm of filters like Gaussian blur, utilized for image smoothening and noise reduction, along with morphological filters like erosion, dilation, and opening/closing, which proved instrumental in merging small noises in the foreground with the background.

As we progressed through the tasks, we encountered various OpenCV functions, including 'connectedComponentsWithStats', which facilitated object segmentation within a frame. Task 3 and 4, preceded by thorough research, provided insights into region IDs, region maps, and the underlying logic, fostering both conceptual and practical comprehension of region-based analysis, alongside the concept of eigenvalues/eigenvectors.

A deeper exploration of the OpenCV library enabled us to familiarize ourselves with its diverse classes and methods, while also attempting to write a few functions from scratch further honed our problem-solving skills and illustrated the tangible impact on our programming and logic implementation. These experiences collectively enhanced our proficiency in C++ programming and fortified our understanding of various functions within the OpenCV library.

With the knowledge we gathered, lots of research, and working hard on programming and fixing errors, we managed to build a system that can detect objects in real-time on a 2D screen. It's a big achievement for us and shows how much we've learned and grown during this project.

5. Acknowledgement:

We want to extend our gratitude to Professor Bruce Maxwell for his invaluable lectures, which served as the foundation for our understanding of various algorithms and their importance in our real-time 2D object detection project.

Additionally, we would like to thank the following resources and individuals for their contributions:

- Medium articles that explained Gaussian blur in simple terms.
- OpenCV's user-friendly guides, which helped us understand how to use the `rand` function and other features.
- Udacity tutorials, which made it easier for us to grasp concepts like region-based analysis.
- YouTube videos that broke down the K-nearest neighbors algorithm for us.
- Websites like Wikipedia and GeeksforGeeks, which gave us extra help with understanding algorithms like K-means.
- Math

We're grateful for all these resources and people for making our project possible and helping us learn along the way.