Project 3 : 2D Object Detection

Created by Ravina Lad, last modified on Mar 09, 2022

Summary :

This project is about building a system program that recognizes 2D objects. . The goal is to have the computer identify a specified set of objects placed on a white surface in a translation, scale, and rotation invariant manner from a camera looking straight down. Also, it should recognize the objects in real time. For the same purpose, processes involved in this object detection program are : thresholding binary image, cleaning binary image and then segmentation of image/video stream, also calculate the feature vector which are invariant to : scale and orientation, and finally the comparing it with our database.

Lab Setup :

Task :

1. Threshold the input video
Thresholding is separating the pixel of an image into 2 classes. That is foreground and background. There are various type of thresholding, I have implemented this task with 3 variations: **(ALL FROM SCRATCH, One method can be considered for Extension)**

1. one - sided threshold : converting an image into gray-scale(converting to gray-scale function written from Scratch) and setting constant threshold value to convert Live video feed into binary.
2. Converting frame to HSV and blurring the image and then : taking saturation value to set threshold.
3. Used single threshold on intensity values, and transformed the pixels little bit. So, the pixels which were very saturated I made them very dark. I sort of came with a multiplier that got smaller when pixel is more saturated. Calculated saturation and intensity without converting image into HSV color space.

2. Clean up the binary image
Clean up Binary implemented also from scratch. In this task, growing and shrinking : morphological operations performed on frame to remove noise and holes. Both operations performed same number of times so that the original size of image won't change. Growing will connect nearby boundaries and shrinking will disconnect them. This is implemented using Grassfire Transform.

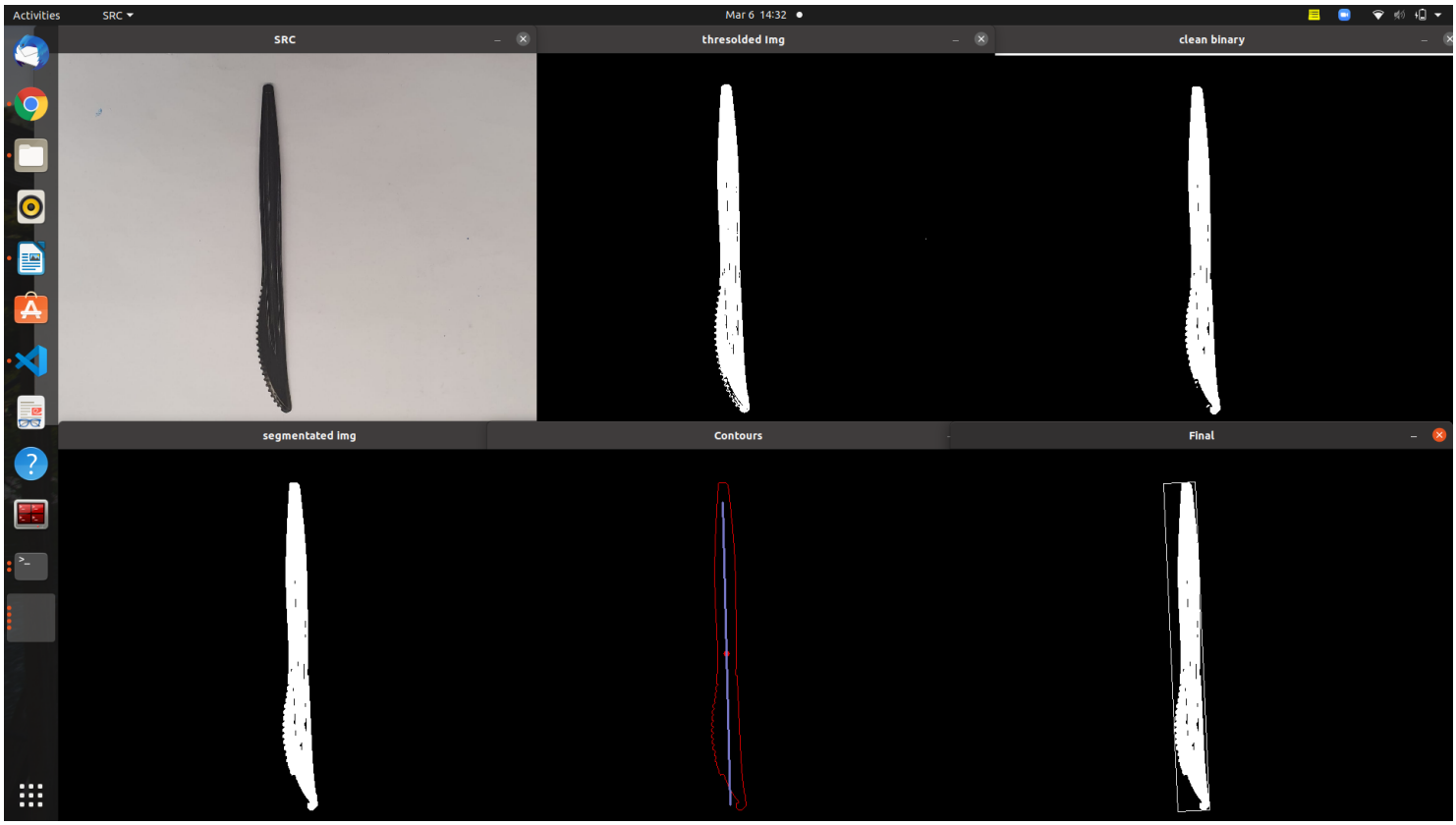
3. Segmentation
1. For segmentation I used connectedComponentsWithStats() function to identify major components in the given binary image.
2. Then used these components to segment the image.
3. This tells us about how to extract label of different regions in an image.

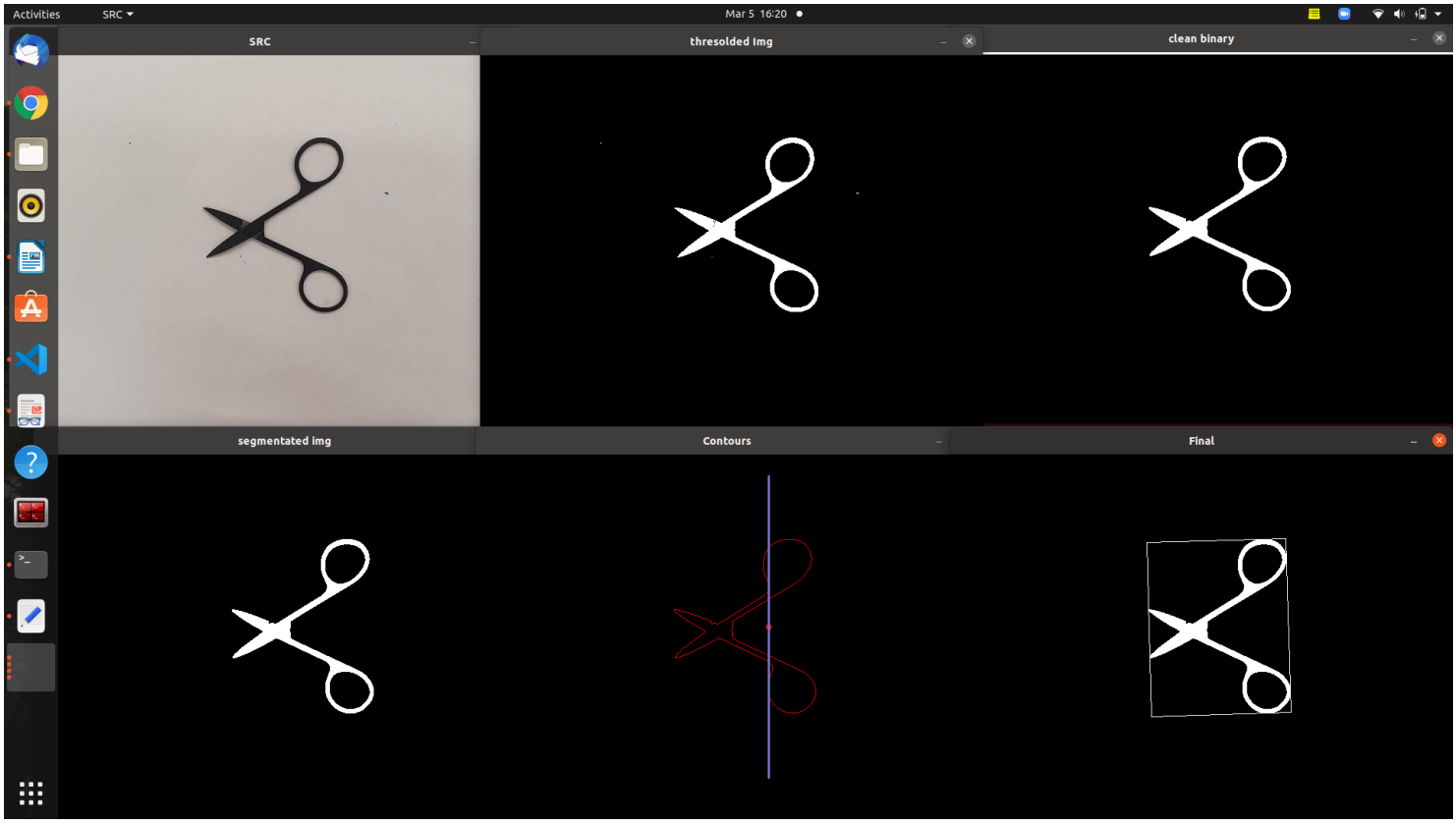
4. Compute features for each major region
1. In this task, I computed different features which are invariant to image transformations.
2. Calculated centroid and oriented bounding box along the central axis, also calculated central axis angle. and used centroid to plot central axis line.
3. Hu moments are moments which are invariant to image transformations. They are a set of 7 numbers calculated using central moments. The first 6 moments have been proved to be invariant to **translation**, **scale**, and **rotation**, and **reflection**.
4. Also, calculated : aspect ration that is height and width ratio and Percentage filled ratio.
5. HuMoments are then log-normalized.

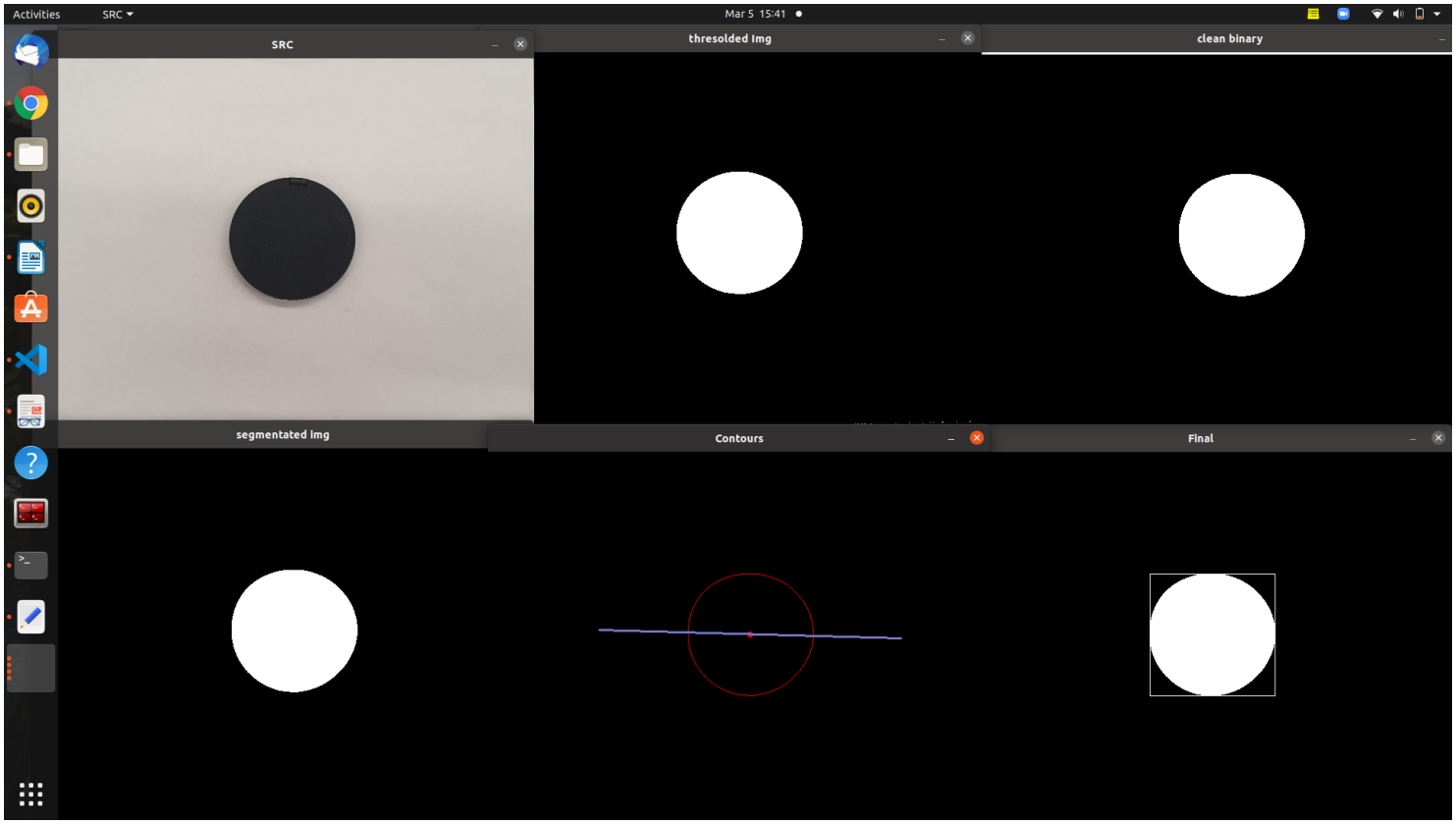
5. Collect training data
1. This task is done from labeled still images of the object, such as those from a training set.

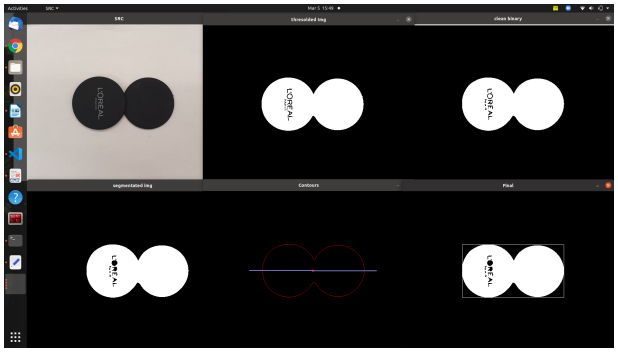
2. Computed feature vectors for all labeled images : HuMoments, aspect ratio and percentage ratio.
3. Stored all the feature vectors into csv file.

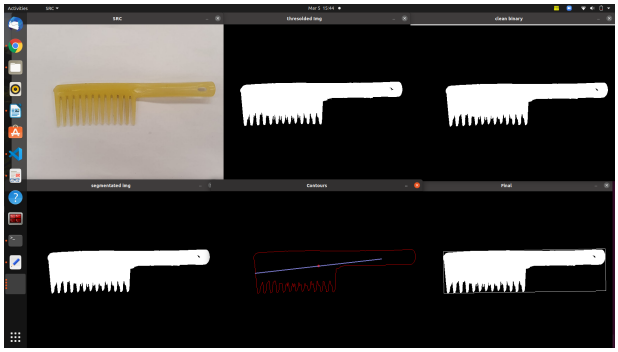
Combined result of all Tasks till now :

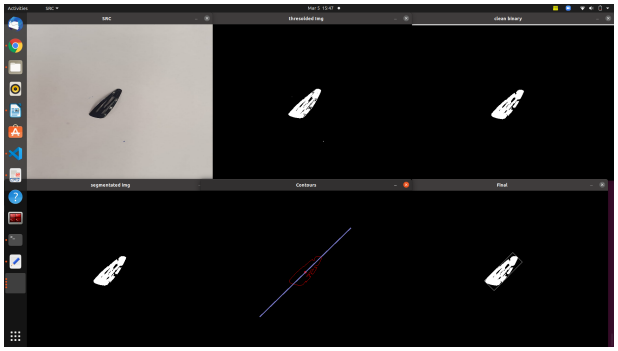


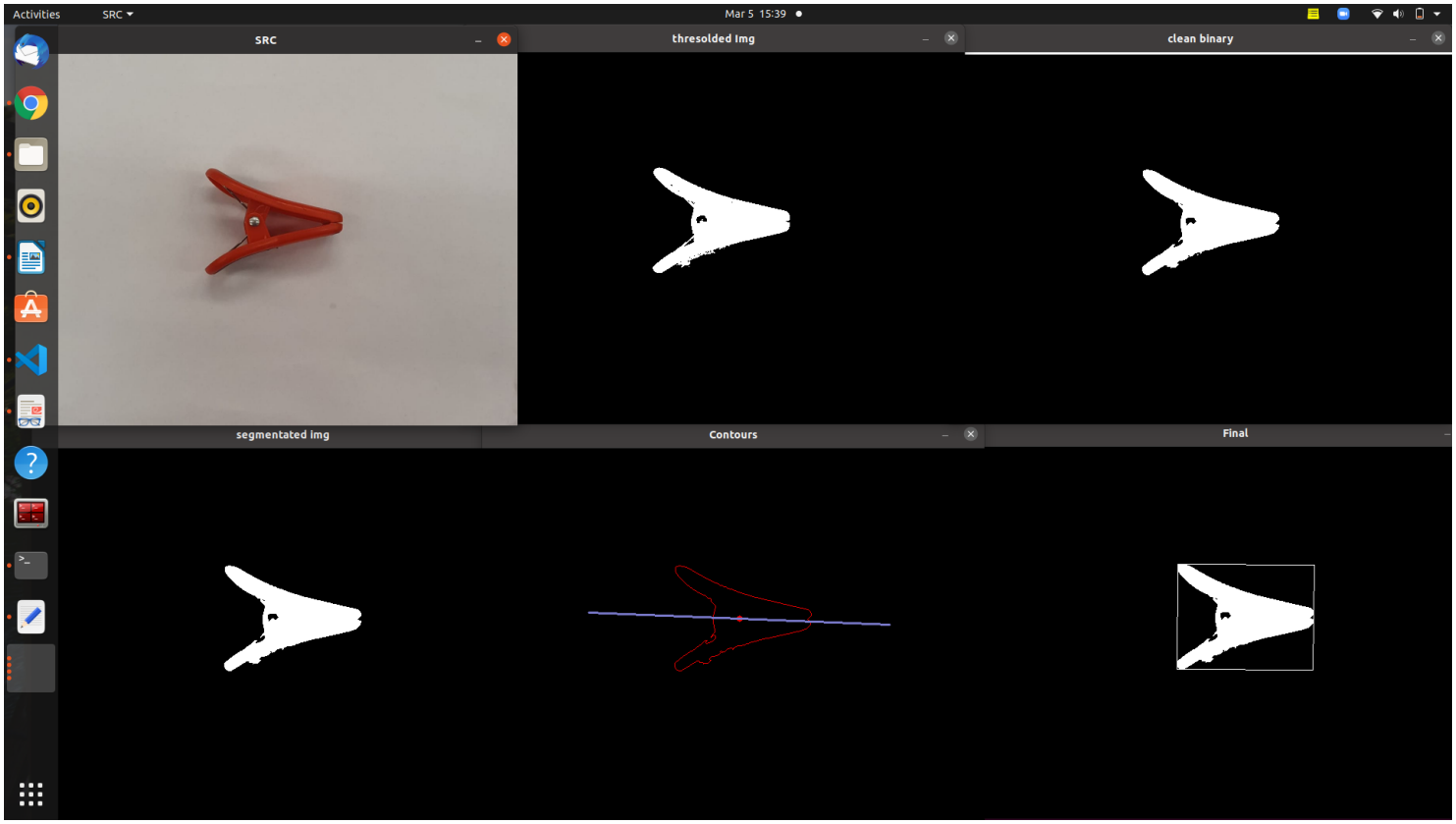


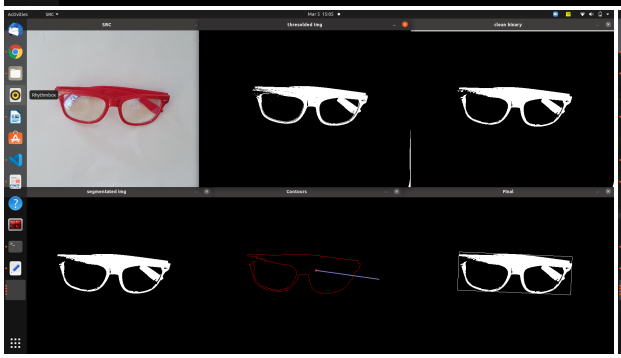

Activities
SRC
Mar 5 15:41
SRC
thresolded Img
clean binary
segmentated img
Contours
Final

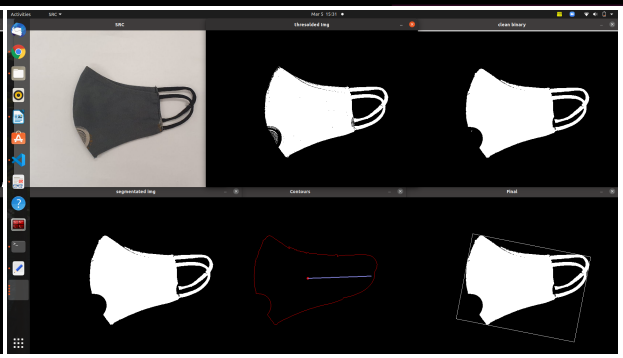


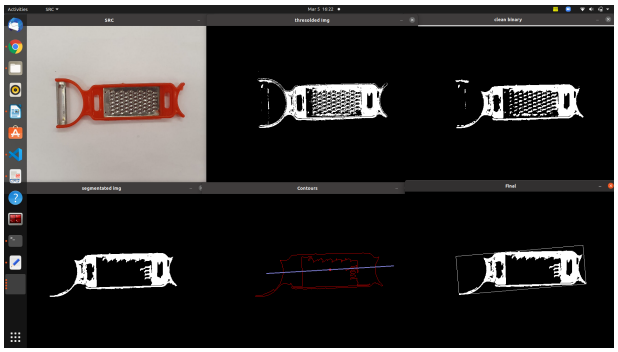


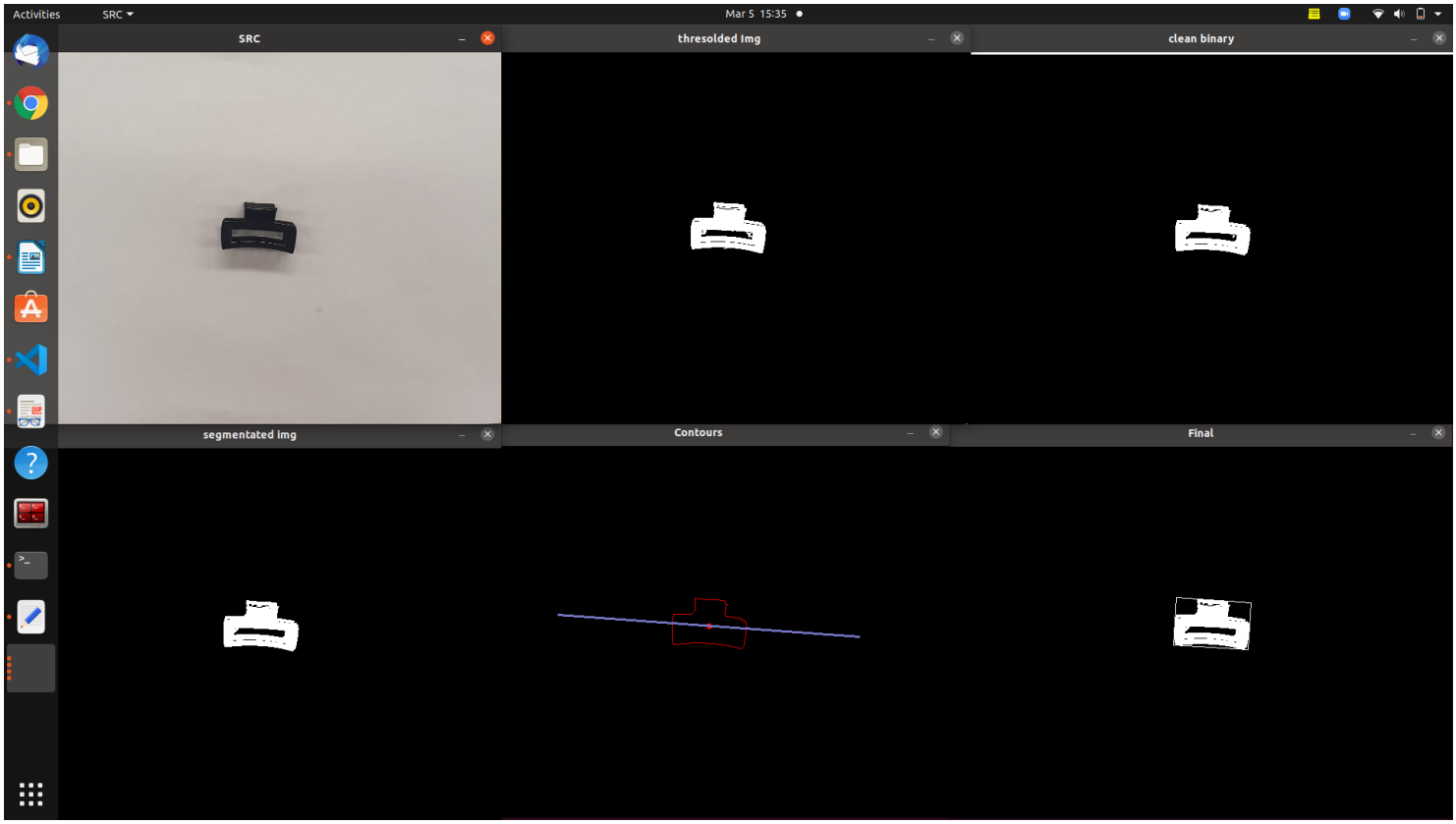


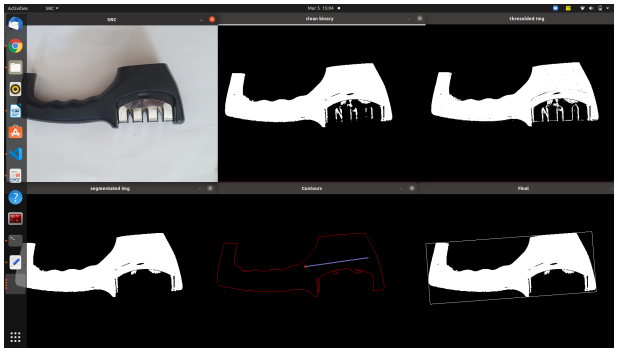

Activities
SRC
Mar 5 15:39
SRC
thresolded Img
clean binary
segmentated img
Contours
Final

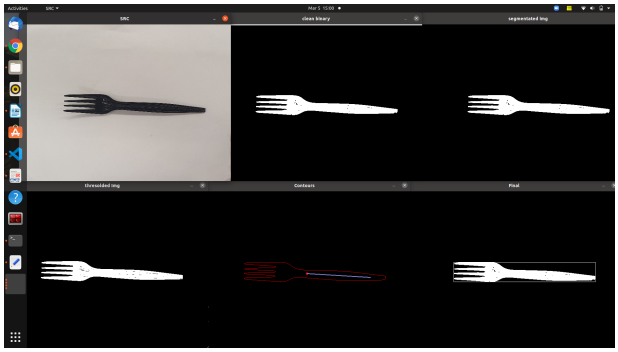


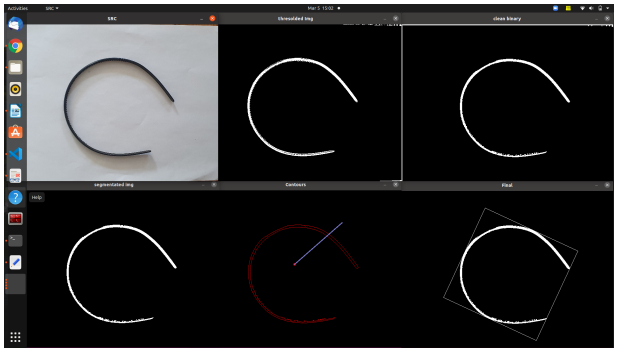


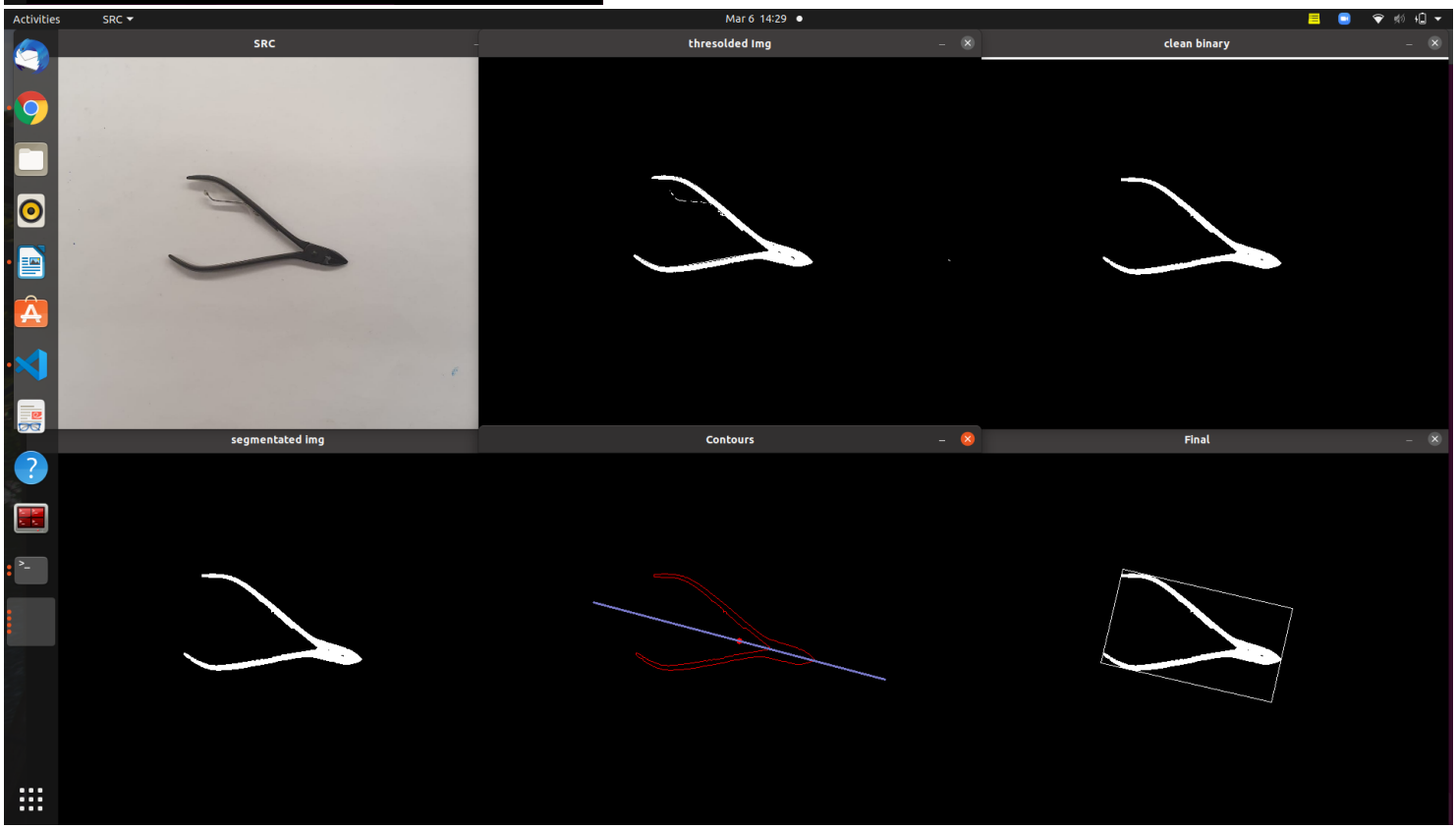



Activities
SRC
Mar 5 15:35
SRC
thresolded Img
clean binary
segmentated img
Contours
Final



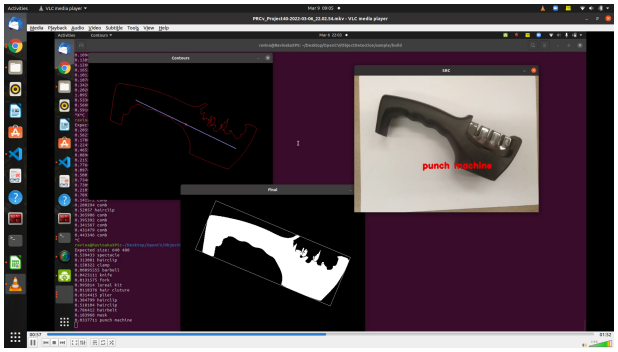


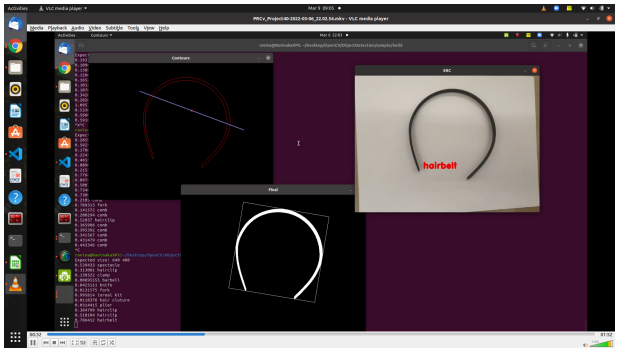


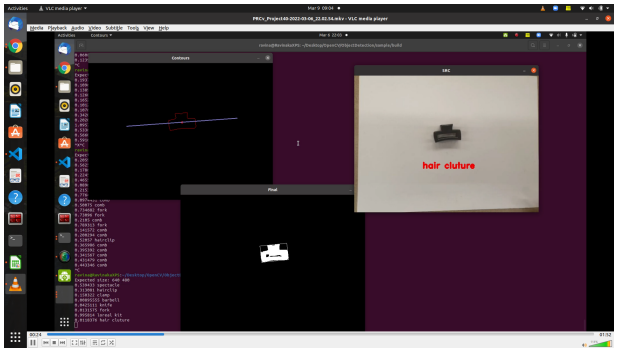


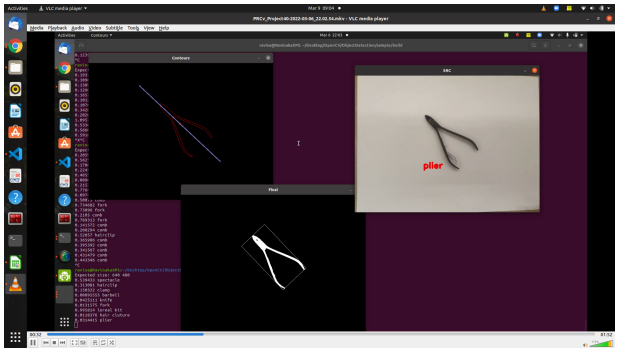
6. Classify new images

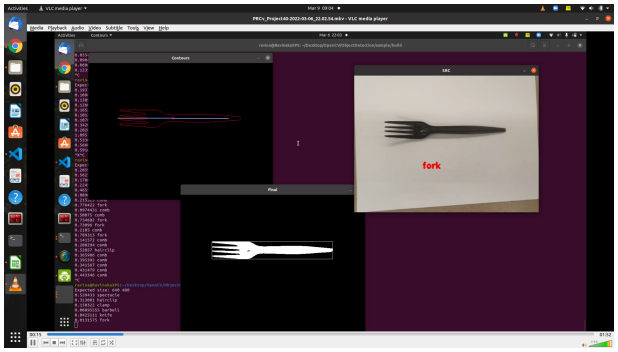
1. Now, that we have set of feature vectors stored in csv file, will read the data.
2. Compute scaled euclidean distance with respect to testing image.
3. Classify the result and put label on source image.

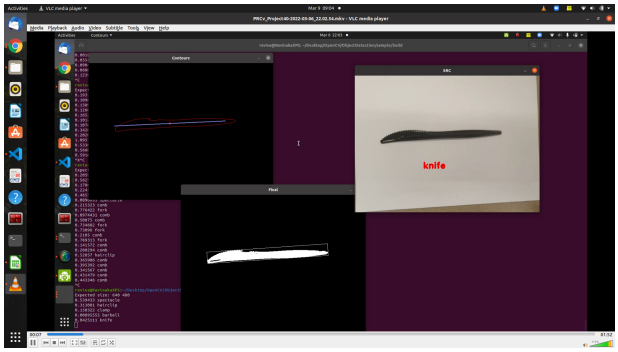


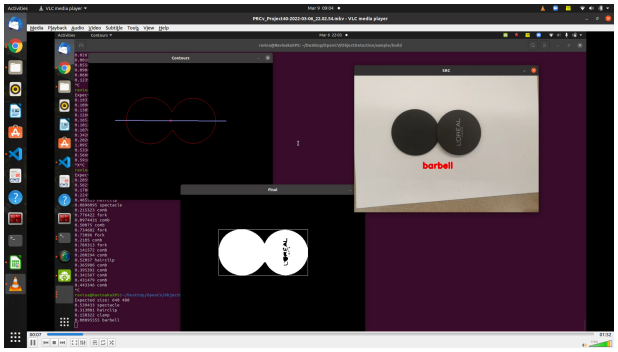


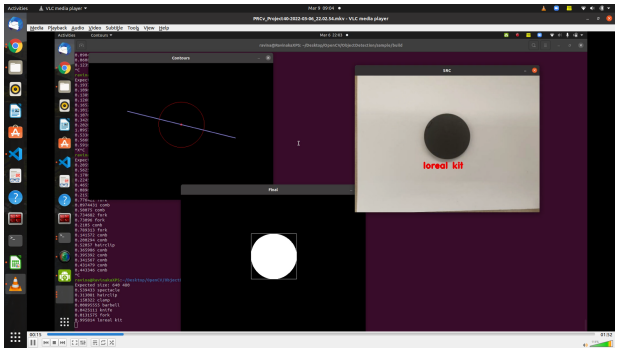


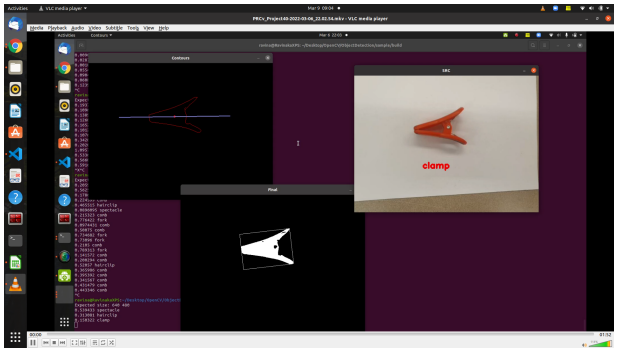












7. Implement a different classifier

- Implemented KNN(K = 1) and k-means clustering (K=3) alogorithm from Scratch. (Also, part of **Extension**)
- When k is small, such as k = 3, one outlier may have an impact on the final result of the k-means clustering model.
- Created a new classifier: k-means clustering that estimates the distance to each feature vector in the database and counts which label class counts the most to better manage outliers.
- The case of underfitting occurs when the value of k is increased. In this case, the model would be unable to correctly learn on the training data.
- The condition of overfitting occurs when the value of k is smaller. The model will capture all of the training data, including noise. The model will perform poorly for the test data in this scenario.

8. Evaluate the performance of your system

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		hairbelt	fork	punch	spectacle	mask	Hair cluture	clamp	Loreal kit	comb	hairclip	barbell	scissor	grater	plier	knife	fan
2	hairbelt	10															
3	fork		8		2											1	
4	punch			10						1				1			
5	spectacle				7												
6	mask					9											
7	Hair cluture					1	10										
8	clamp							8									
9	loreal kit								10								
10	comb									9							
11	hairclip										10						
12	barbell											10					
13	scissor												9		1		
14	grater													9			
15	plier							2					1		9		
16	knife		2		1											9	
17	fan																10

9. Capture a demo of your system working

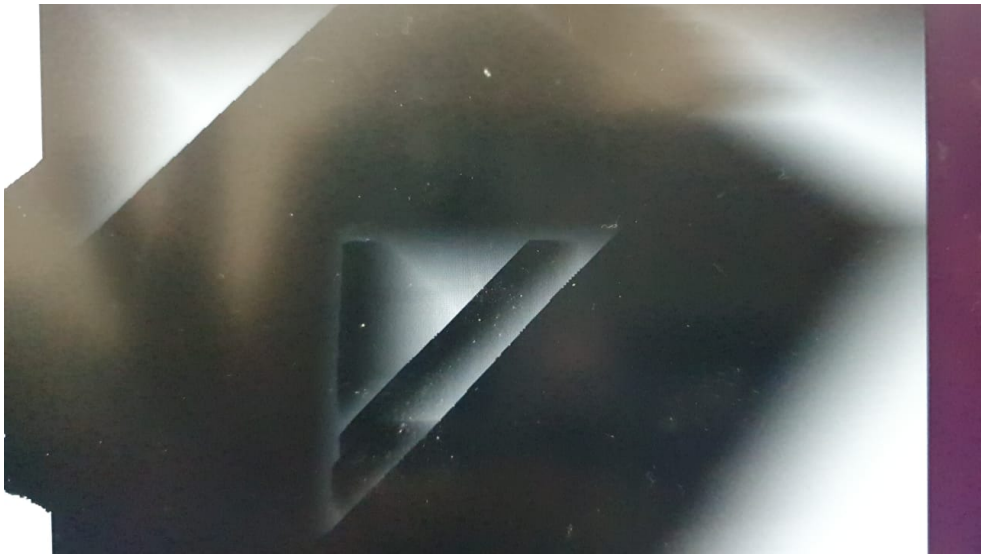
Object Detection

Extension :

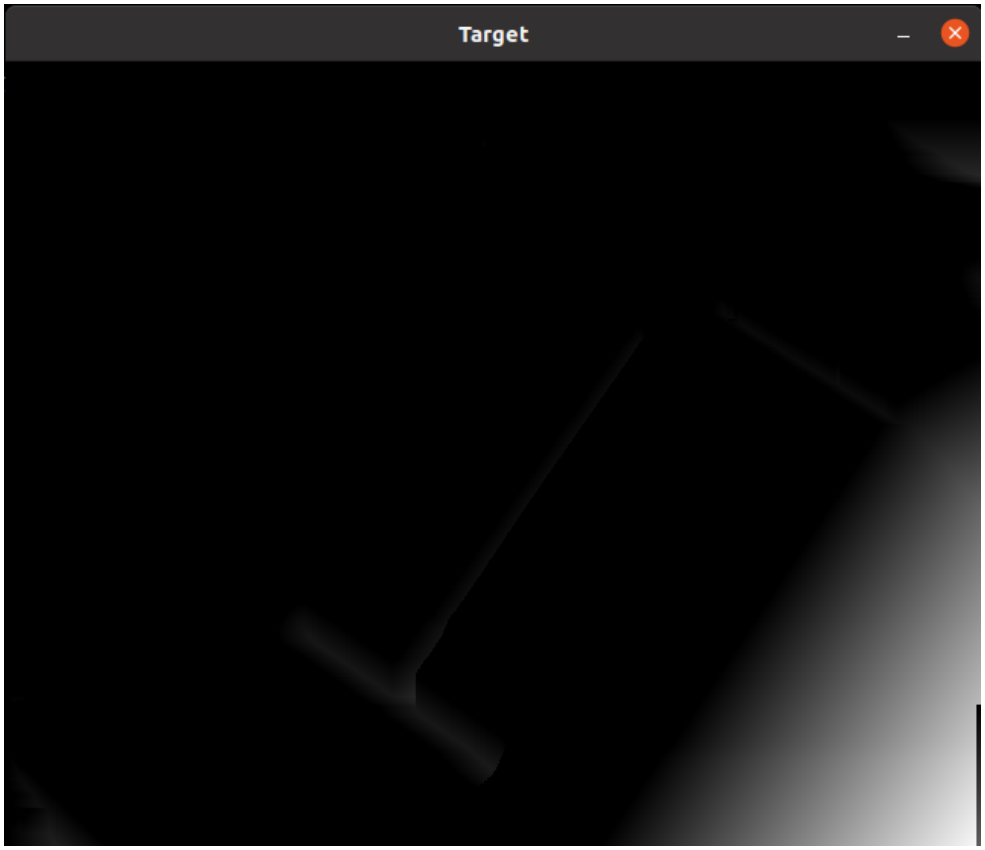
1. GrassFire Growing and Shrinking (Scratch):

1. Implemented Grassfire Growing and shrinking from scratch.
2. The **Grassfire transform** is the computation of the distance from a pixel to the border of a region.
3. This implemented calculating manhattan distance of foreground from background.
4. We can give count to function to set the number of growing and shrinking.

Growing Grassfire Tranform :

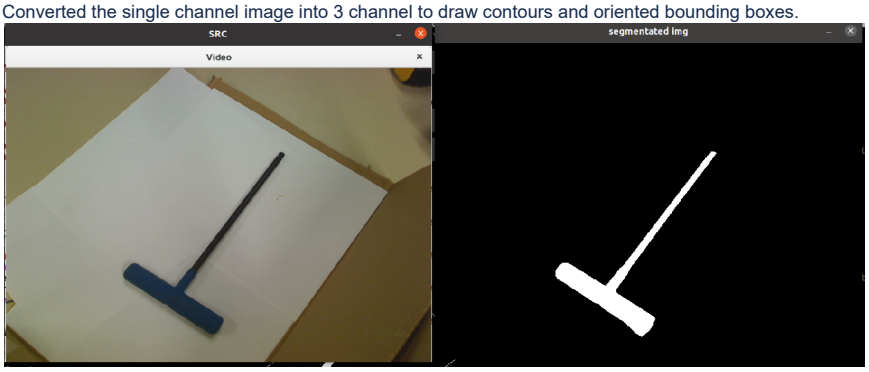


1.
2. Shrinking Grassfire Tranform :

3. 

2. Segmentation (From scratch):

- Used connectedComponentsWithStats stats and label and set area threshold
- Then, compared all areas and their labels depending on the threshold area value
- Converted the single channel image into 3 channel to draw contours and oriented bounding boxes.



3. More than 10 objects:

I considered around 16 images for this task of 2D object detection.

4. Comparison of different distance metrics on different classifier k-means clustering:

- I have implemented 4 distance metrics from scratch without using any libraries.
- Implemented to check which one works best for object matching.
- Manhattan Distnace L1 Norm
- Chi-sqaure
- Correlation
- Scaled Euclidean Distance
- After trying out above distance metrics, accuracies of above distance metrics given below :

	A	B	C	D
1	Type of Distance Metric	Detected Objects	Total Objects	Percentage Accuracy
2	Manhattan L1 Norm	13	16	81.00%
3	Chi-square	12	16	75.00%
4	Correlation	9	16	56.00%
5	Scaled Euclidean	15	16	94.00%

Reflection :

- Learned about the fundamentals of image identification which include not just texture and color but also scale, rotation, and position.
- Learned different region properties Moments, centroid, central moments, central axis angle, HuMoments, Bounding box size, Percentage filled, and oriented bounding box.
- Various types of moments, such as central and hu, were learnt, and the learning process developed a good comprehension of these ideas and what kind of information they maintain.
- Learned how to improvise inbuilt opencv functions and write better implementation from scratch.

Acknowledgement :

I'd like to thank the sites listed below for their assistance in understanding and referencing various concepts while working on this project. I'd also like to express my gratitude to Professor Bruce for his great video explanations of these ideas. I would also like to thank my classmates and friends Dhruvil Parikh.

https://learnopencv.com/shape-matching-using-hu-moments-c-python/

https://answers.opencv.org/question/8871/how-to-calculate-humoments-for-a-contourimage-in-opencv-using-c/

https://answers.opencv.org/question/120698/drawning-labeling-components-in-a-image-opencv-c/
https://docs.opencv.org/3.4/d9/dde/samples_2cpp_2kmeans_8cpp-example.html

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

https://docs.opencv.org/3.4/d8/dc8/tutorial_histogram_comparison.html

No labels