CS5330: Pattern Recognition and Computer Vision - Project 3 Report

Project 3: Real-time Object 2-D Recognition

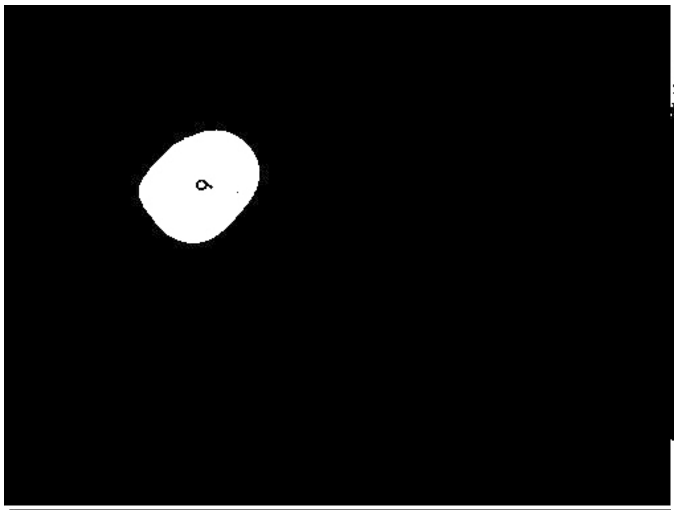
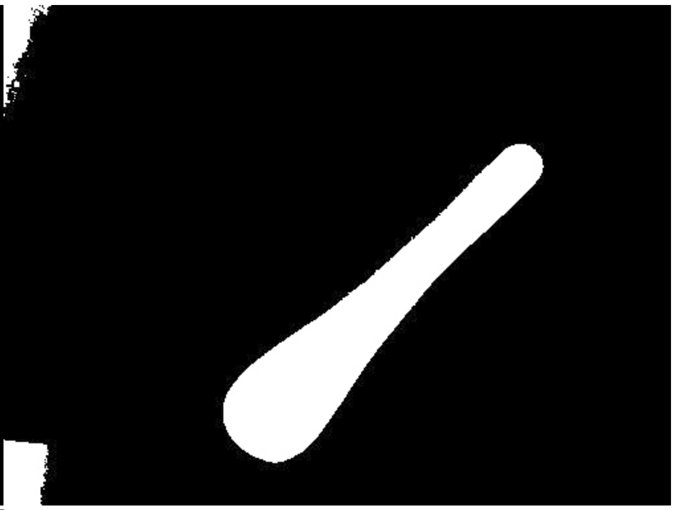
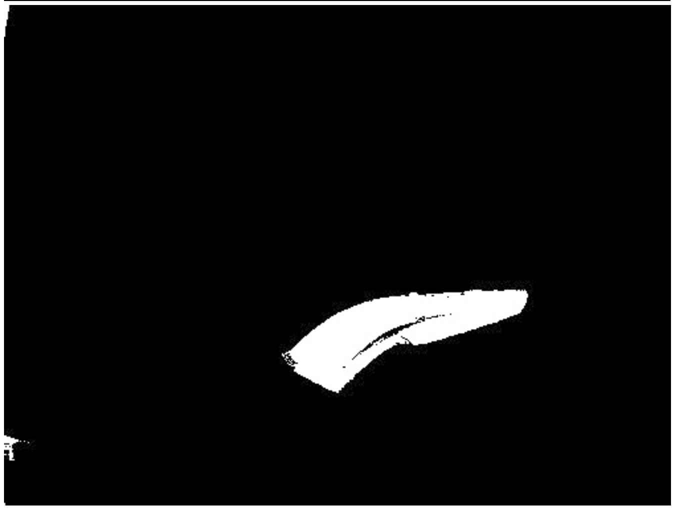
Brief Description of the Project:

This project is about real-time 2D object recognition. The goal of this project is to have the system identify a specific set of objects placed on a light/white surface in a manner that is translation, scale and rotation invariant from a camera that is looking straight down. The system should be able to recognise single objects placed in the image and identify the object as an output image. It should be able to provide a video sequence and be able to do this in real-time. The approach to do this would be to first threshold the input video. It is arguably the most important step of the project as it converts the image into a binary image separating the foreground and background. Nothing could possibly be achieved without fine-tuning the threshold values so as to get an excellent thresholded image. Any amount of good thresholding would still require a little cleaning up which can be done through morphological operators. Then comes the segmenting part which allows us to look at the region of interest discarding all other regions. Later on, we go on to compute the central axis, draw an oriented bounding box, and compute features that are rotation, scale and translation invariant as asked. Then the system is trained on a set of objects and new objects are classified with the help of it. Results are also checked on another classifier and a confusion matrix is built out of it. Let's take a deeper look at each and every step that completes this project along with some amazing extensions.

Original Images:

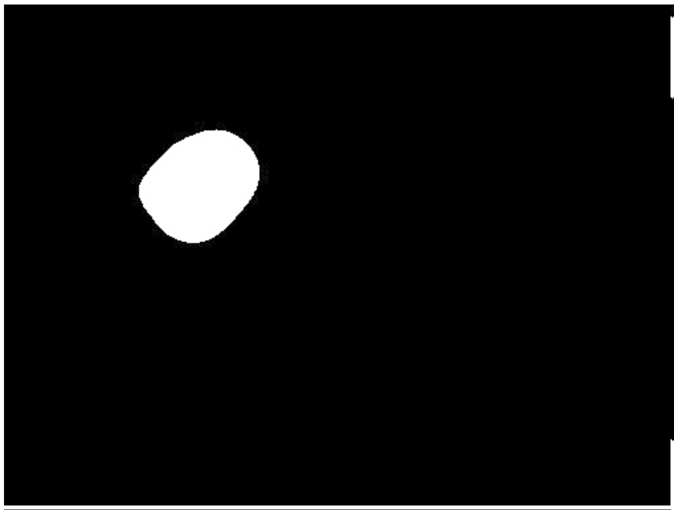
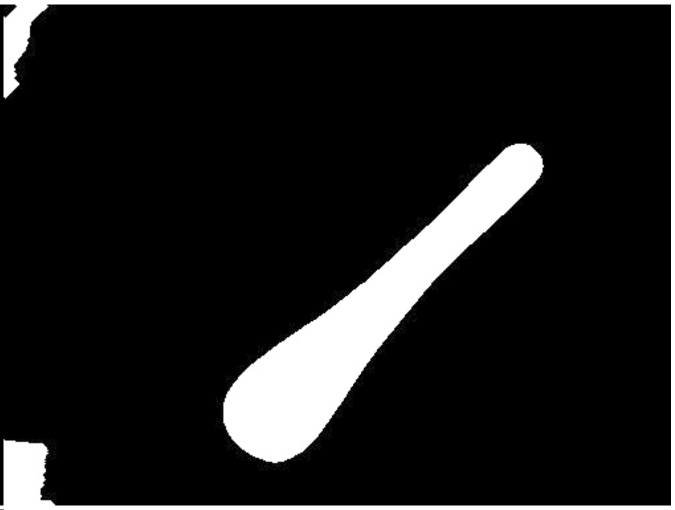
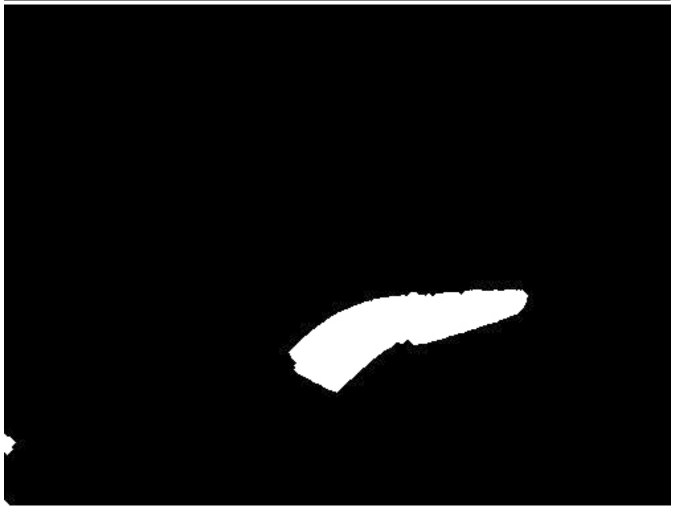
I have a total of 12 objects in my database data.csv out of which these three have been used to meet the requirements of required images. Also, the knn(k = 2) classifier and the kmeans(k = 3) algorithm implemented as an implementation uses only these three images in their database knn.csv and kmeans.csv

Required Images Task 1: Threshold the input Video

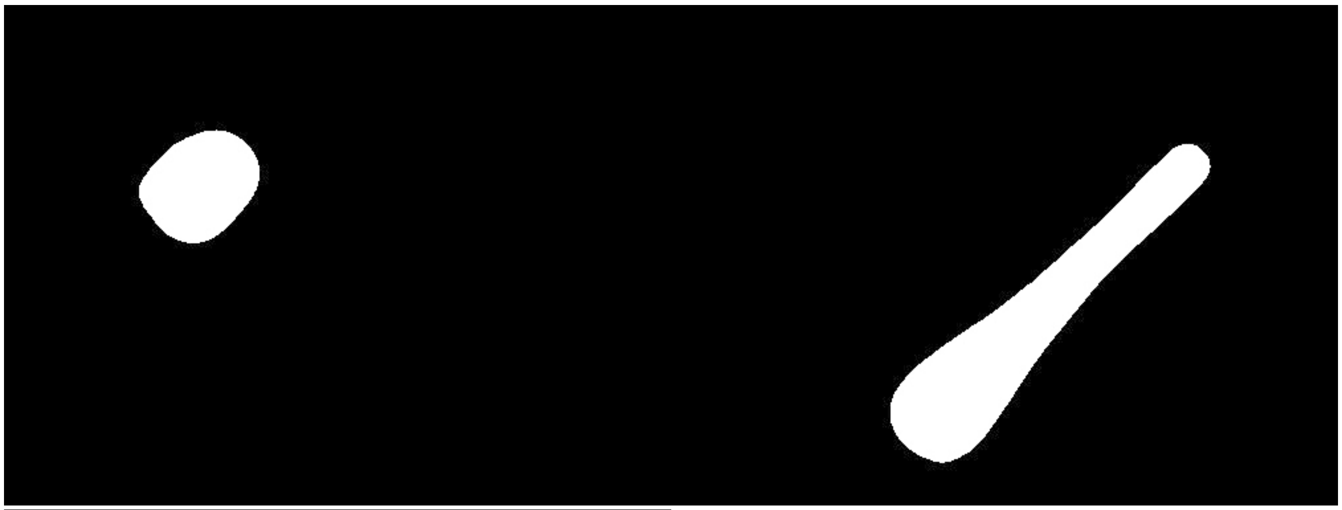
I have implemented two thresholding algorithms from scratch. One of them thresholds on the BGR values and the other converts the input image to HSV thresholding on Saturation and Value components. I found that HSV works better when it comes to picking colours and as I needed to set the white color to the background and the rest to the foreground, the later algorithm works much better than the first one which is what these images have been thresholded with. Of course, this is the feed of an input video. We can observe that shadow has a negligible effect as we are using the HSV color space. The values have been fine-tuned for the setup but have also been tested in different intensities of lights and give good results.

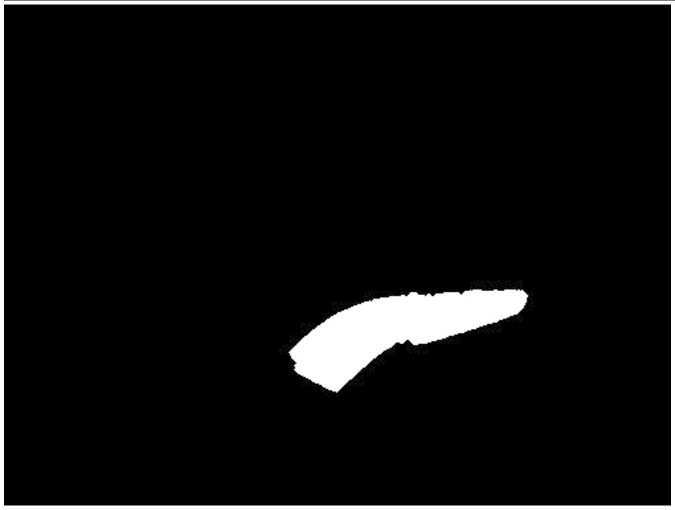
Required Images Task 2: Clean up the Binary Image

After thresholding (we can observe there are still some spaces that require cleanup), morphological operators like growing and shrinking have been used to clean up the image. It takes care of salt and pepper noise (if it exists) while also filling up the holes that might have been a part of the object because of the specular reflection or the body reflection. I have done a number of shrinking and growing operations to clean up the image. I have implemented my own functions from scratch to perform growing and shrinking.

Required Images Task 3: Segment the images into regions



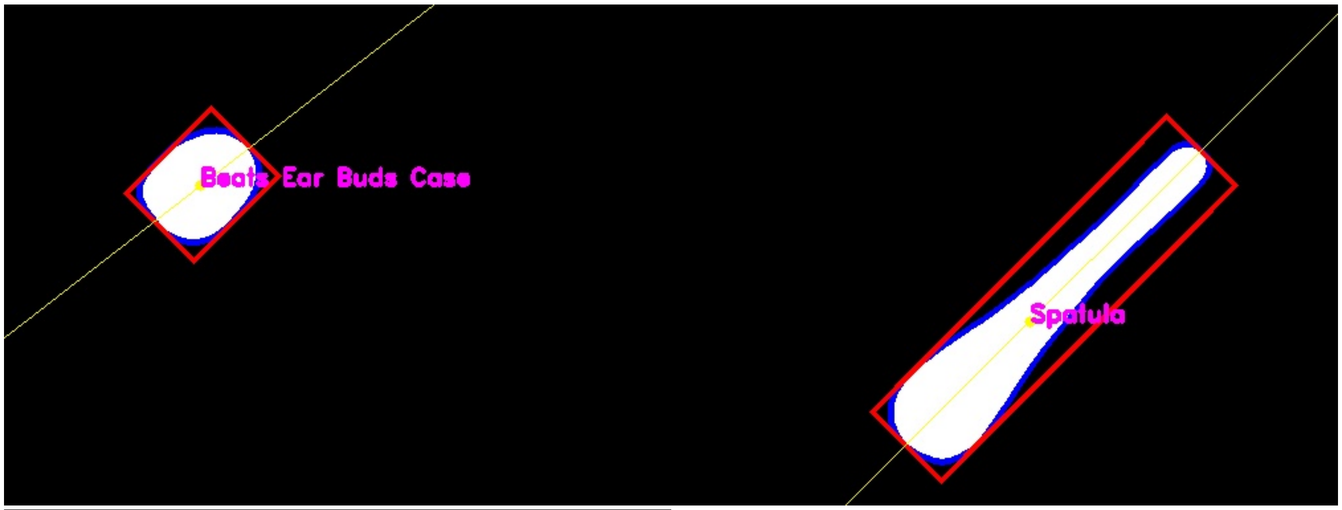


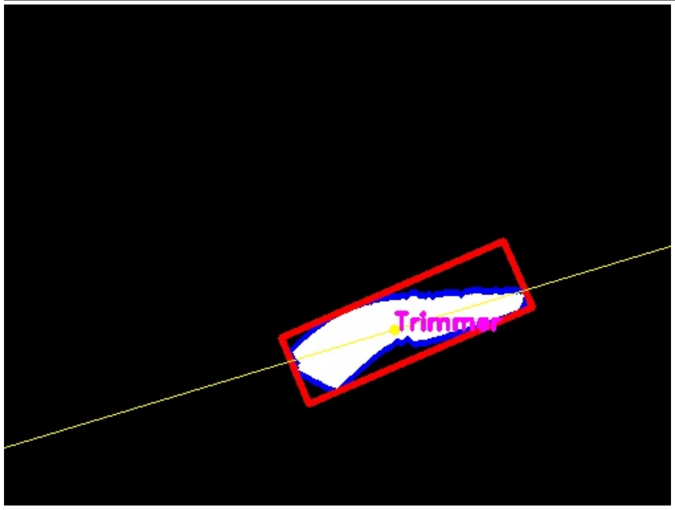
As we can observe from the images of the previous task that even after cleaning up the image, there are still some areas that are a part of the foreground but are not the regions we are interested in. Our regions have not been divided and labelled as well. For this, it is necessary to segment the image. I was initially using the connectedComponentsWithStats function of openCV but it wasn't doing what I wanted it to and so after passing the cleaned up image to this function, I have implemented my own segmentation function as well which takes care of the noisy areas and gives us the region of interest.

It takes care of the noise on boundary regions and uses area threshold to remove unnecessary foreground regions.

It can also detect multiple regions of interest at once as we will see in the Extensions section.

Required Images Task 4: Compute features for each major region





I am using findContours function on the segmented image to get the contours for the object/s. Using the contours (in blue), I am calculating the rectangle with minimum area, extracting those points, and drawing a rectangle using those four points which gives me the oriented bounding box (in Red). After that, I am calculating moments using the inbuilt moments function from contours and finding the centroid using those values (dot in yellow). With the mu components of the moments function, I am calculating the angle of the central axis as seen in class and displaying the central axis (line in yellow).

The features that I am computing to append to my database are aspect ratio, percentage-filled, hu-moment values making it a feature vector of 8 values.

Training Mode Task 5: Explanation

My model launched into the training mode with the keypress n. It asked the user to input the exact path of the image or the frame from the input video feed. It asked the user to give the name of the .csv database file in which they wanted the model to write the feature vector with the associated label and also asked the user if they wanted to reset the database file meaning delete all the data and start writing again. The csv_util header files provided by the professor for Project 2 has been utilised. For that matter, the skeleton code provided for capturing the video for Project 1 has also been utilised.

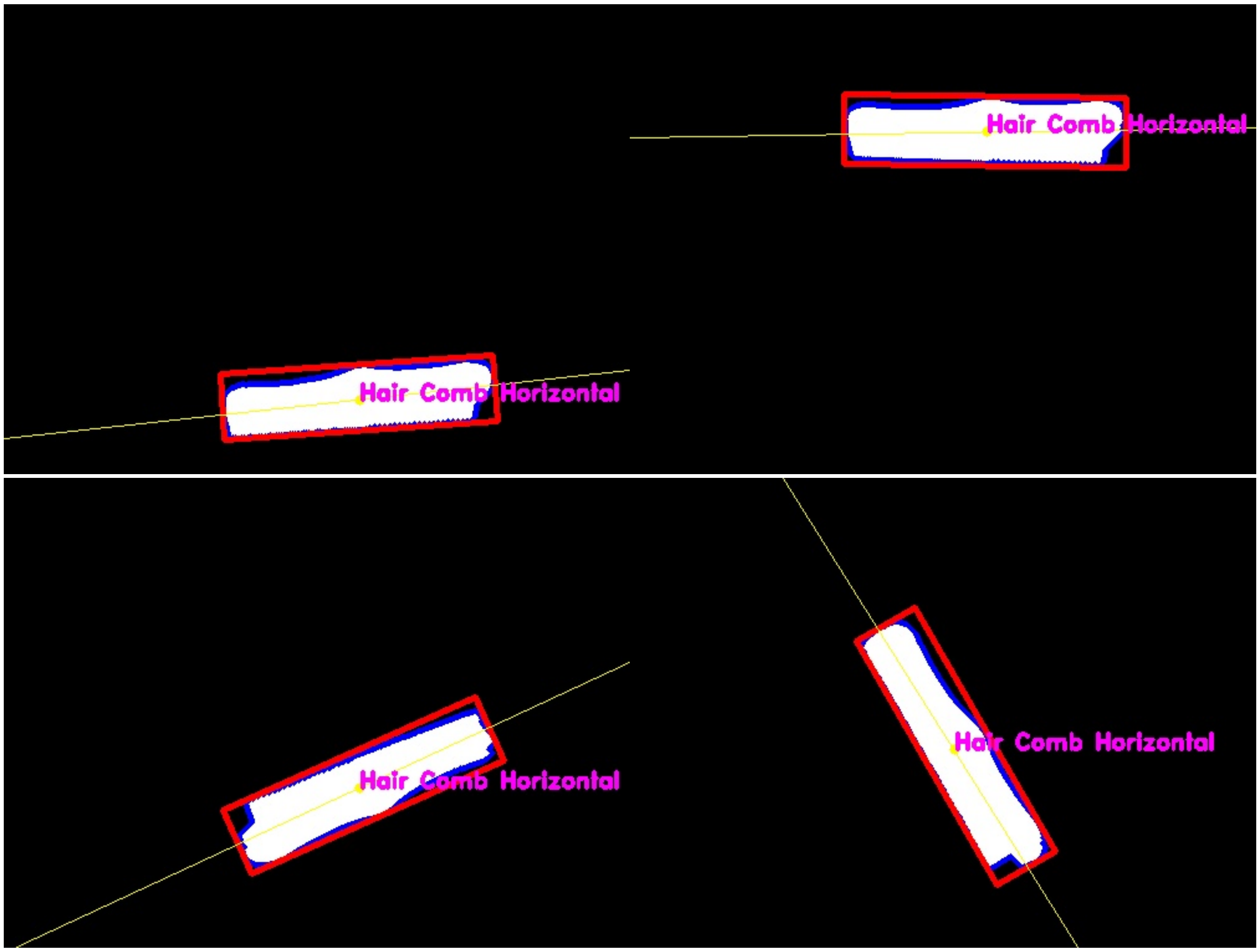
At a later stage, when I was recording the video using droidcam, I had to make some hardware configuration changes and configuration changes in the MOK Ubuntu 20.04 due to which my keypress was affected and was glitching. Hence, as a last-minute effort, I have provided a separate Training mode file that can also be used to train and store feature vectors in a particular database.

Required Images Task 6: Classify New Images

I have 12 images in the database (2 of which are added as an extension for detection of more than 10 images) for classification using scaled euclidean distance as the distance metric. Below are some of the images that are classified using the previously trained model. It is noteworthy that it is able to detect an object in real-time from an input video feed irrespective of the translation, orientation or scale which means that the model is successful in calculating features that are invariant to translation, rotation and scale.

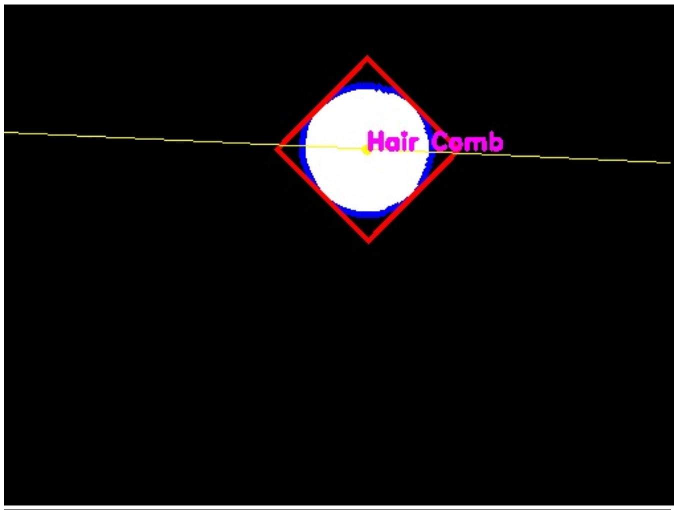
The model is also capable of classifying multiple objects at once as will be seen in the extension section.

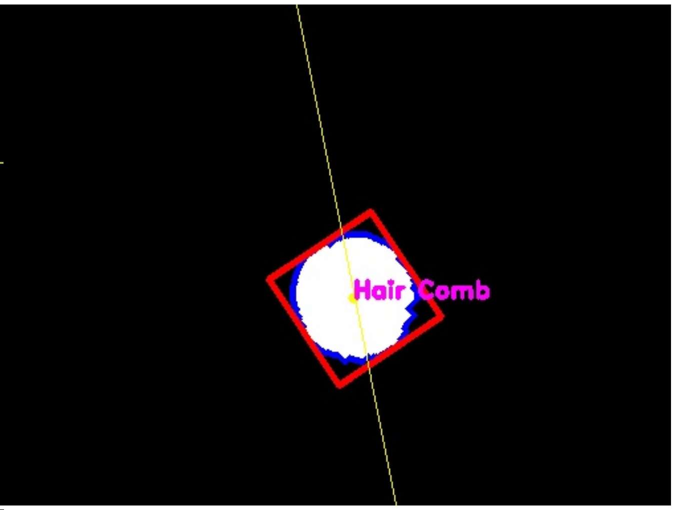
Object 1: Hair Comb Horizontal

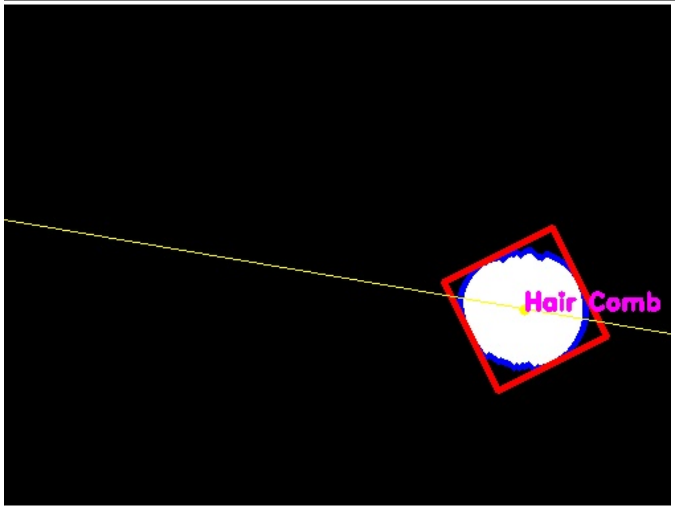


As can be observed from the above images that the hair comb horizontal is identified irrespective of its position, orientation or scale.

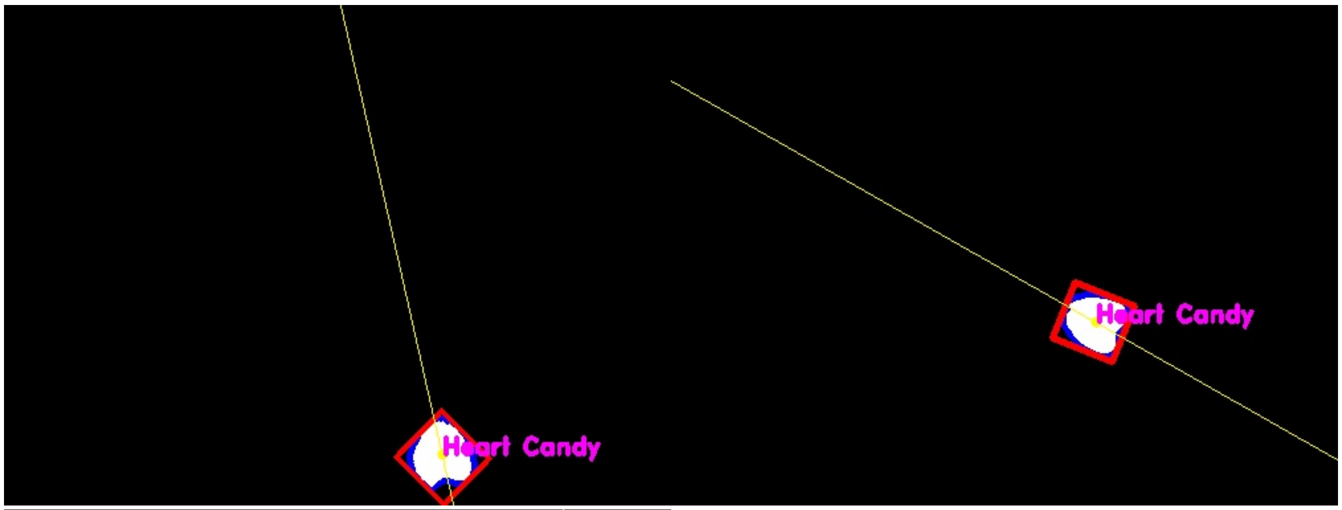
Object 2: Hair Comb

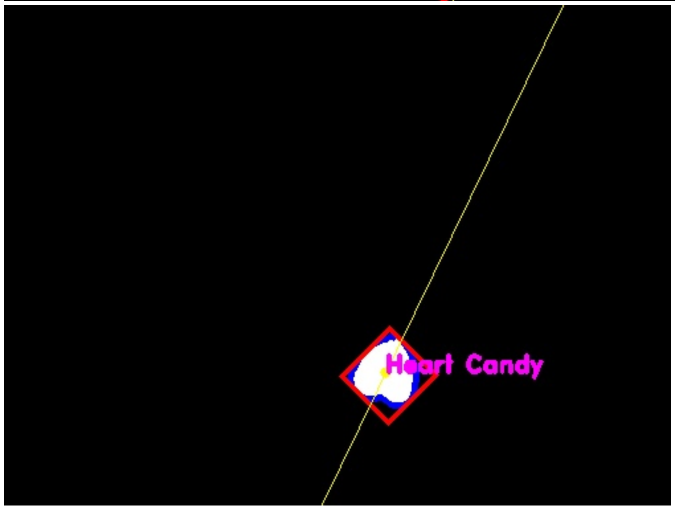




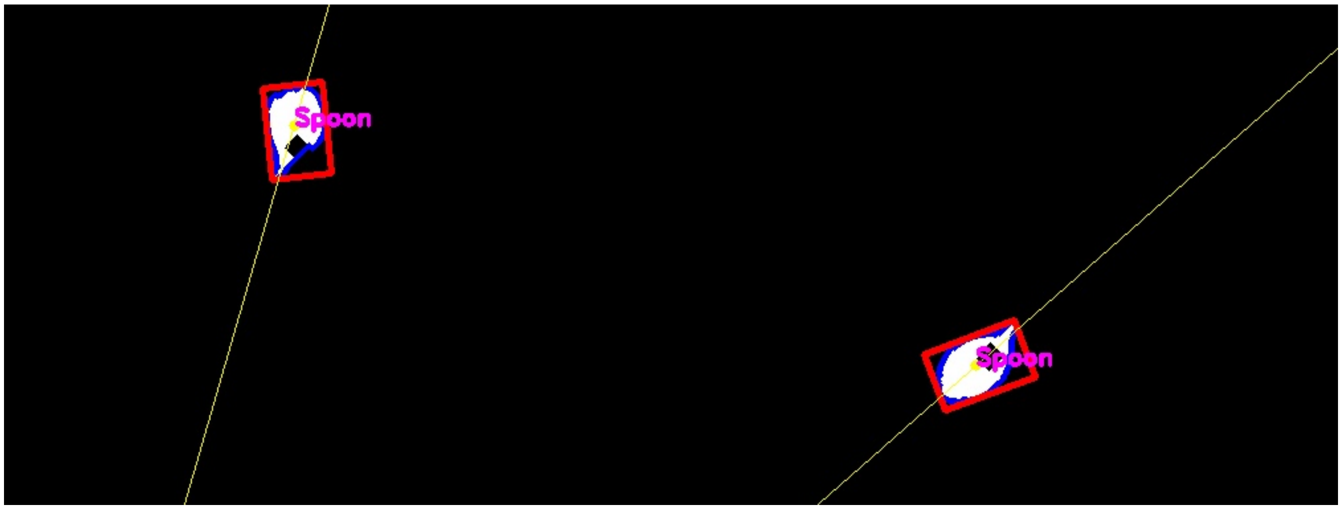


Object 3: Heart Candy



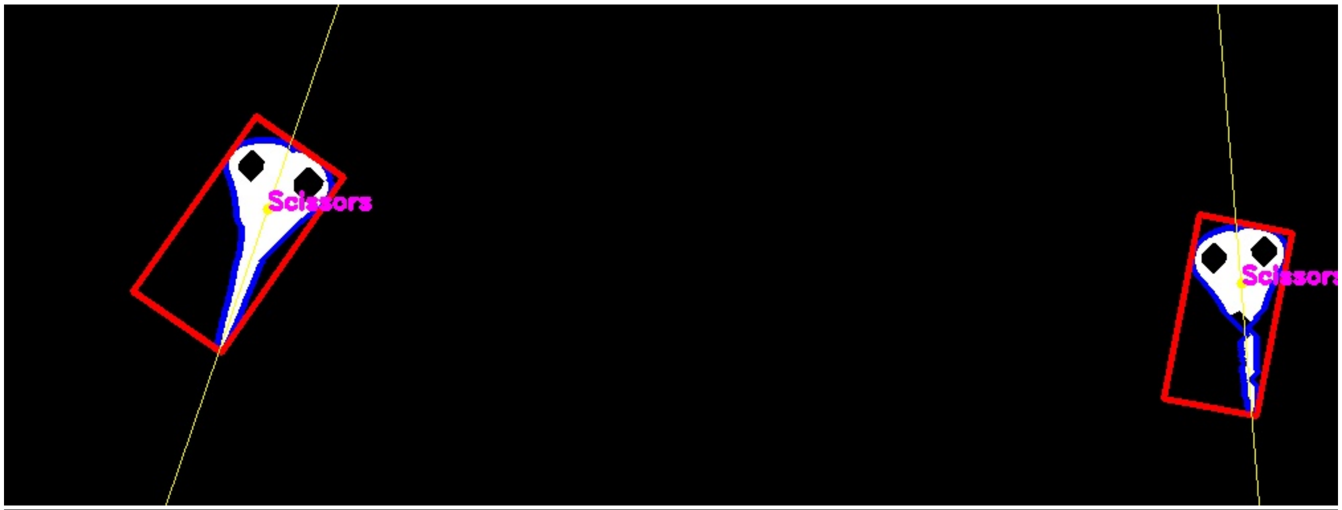


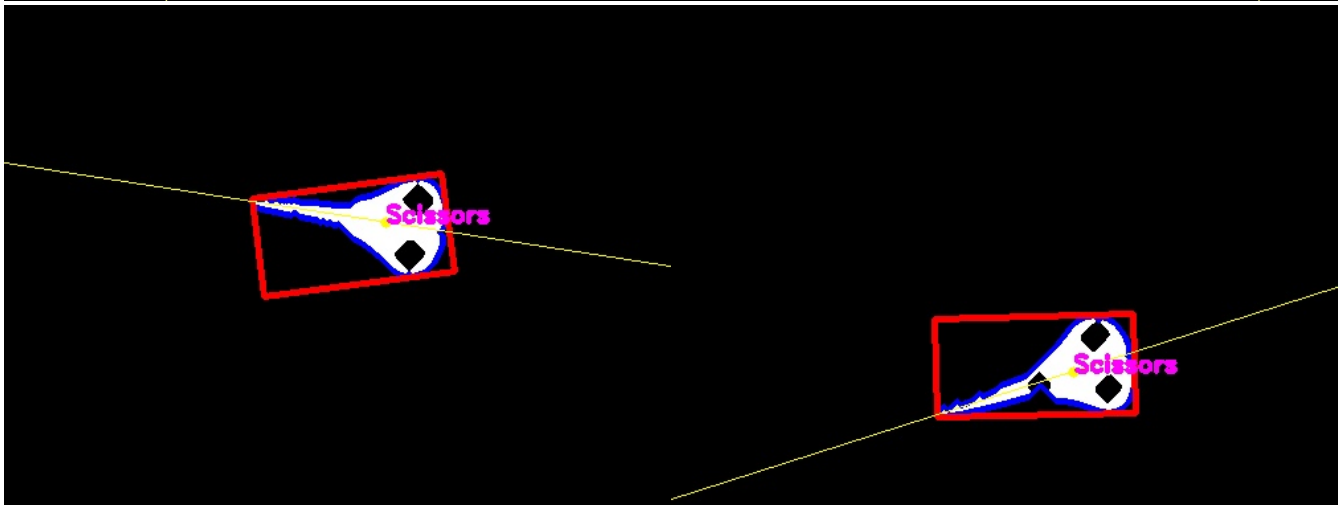
Object 4: Spoon



Because of reflection, the handle of the spoon is completely illuminated (metallic spoon) but it is still able to classify the object.

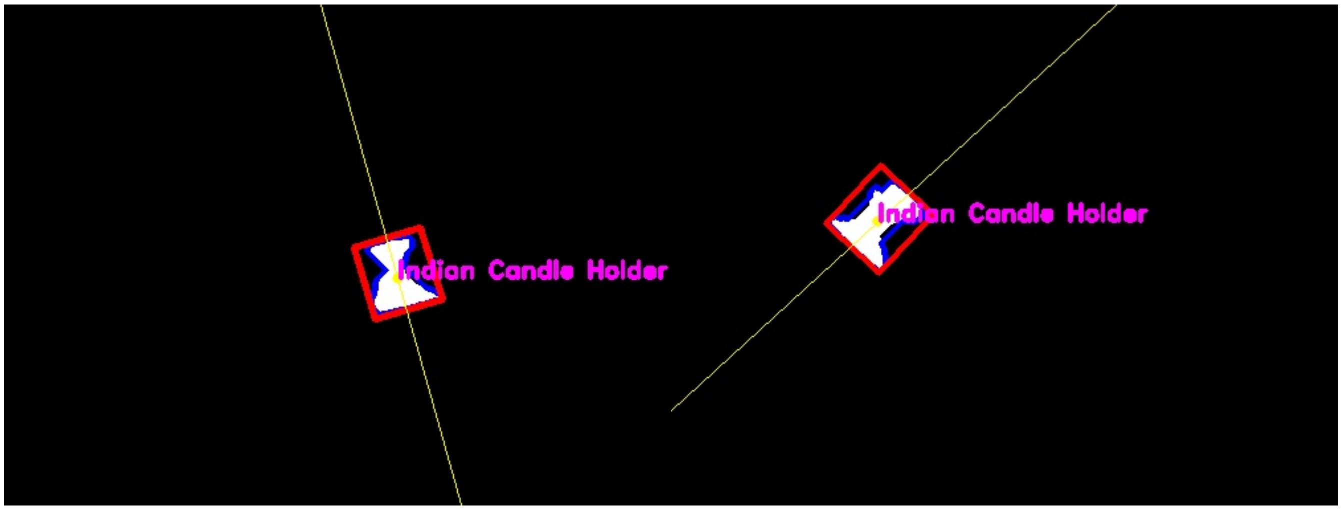
Object 5: Scissors



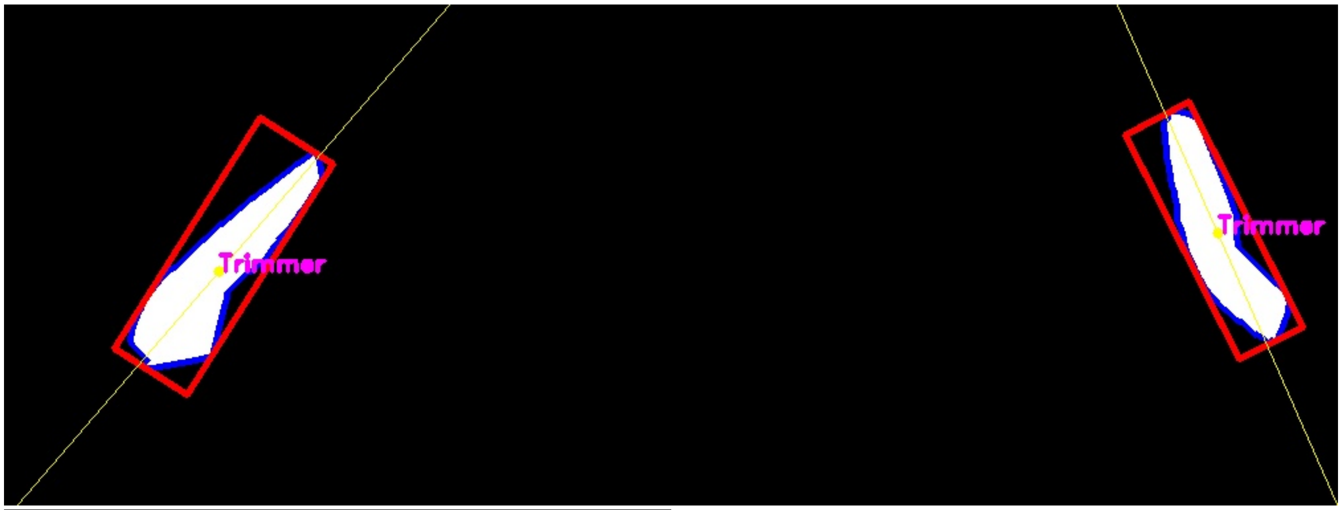


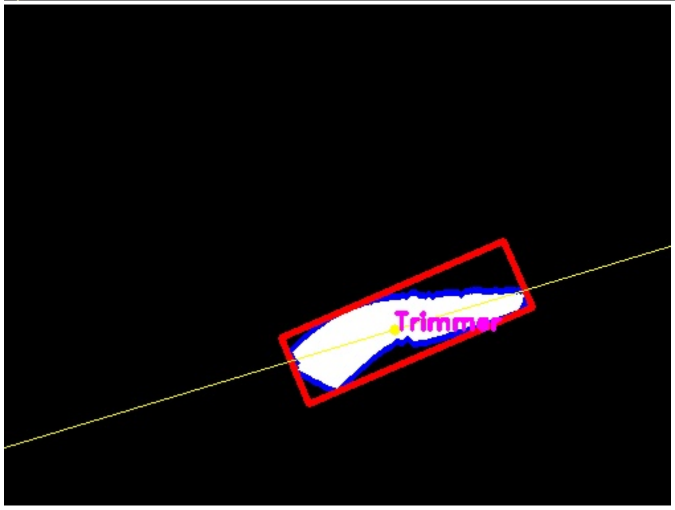
I couldn't help but put a lot of images of classified scissor as it Is a metallic object but the threshold in HSV space compensates for the illumination and reflection and gives amazingly clean images.

Object 6: Indian Candle Holder

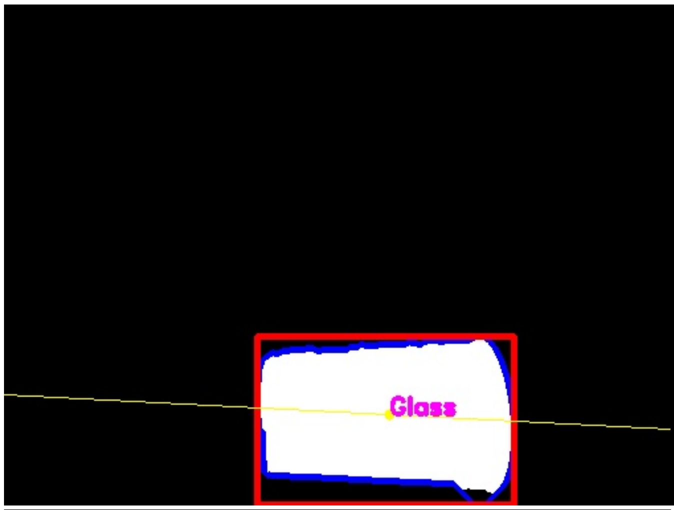


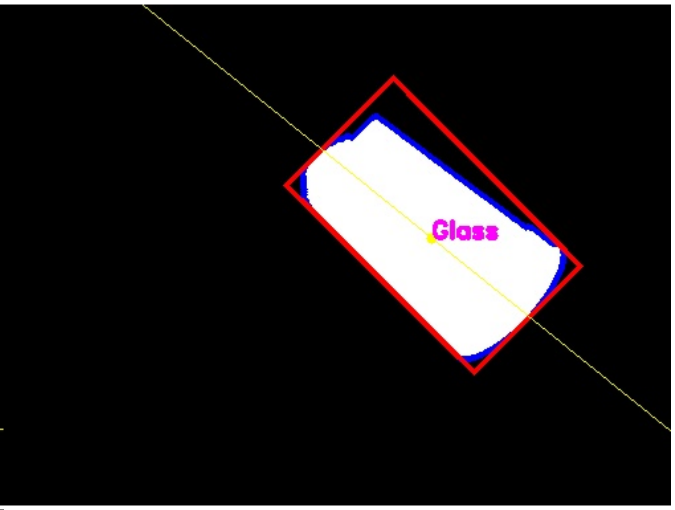
Object 7: Trimmer

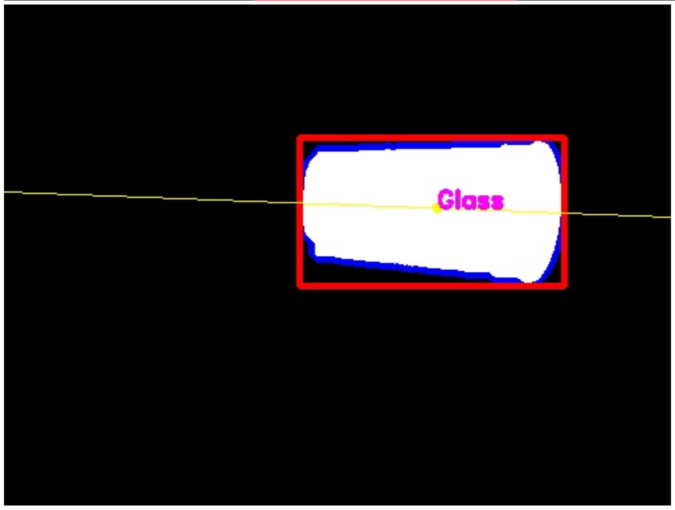




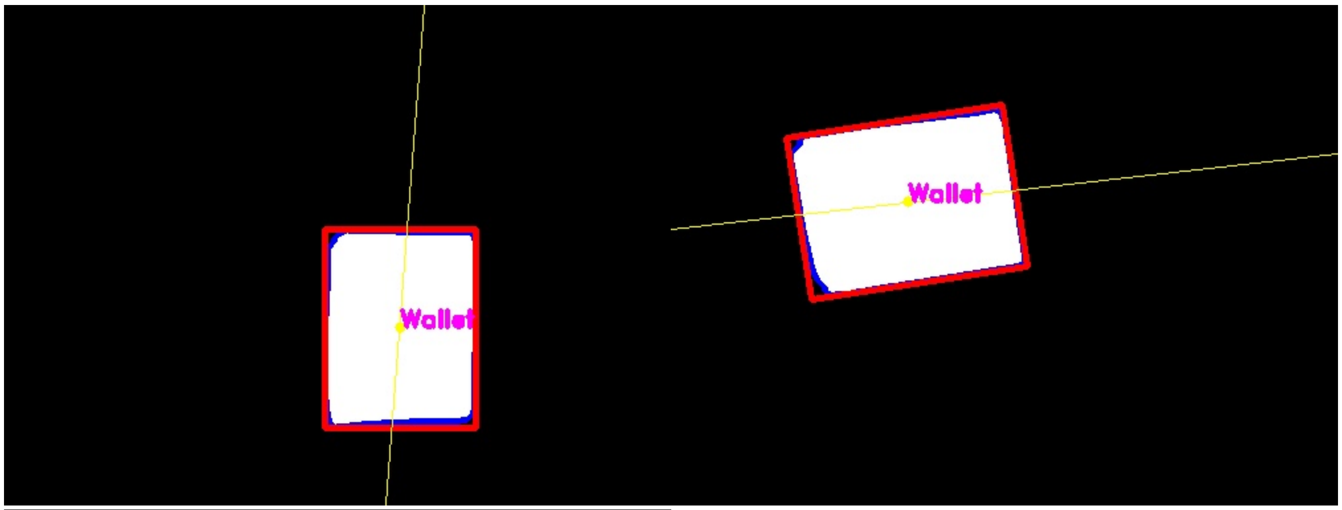
Object 8: Glass

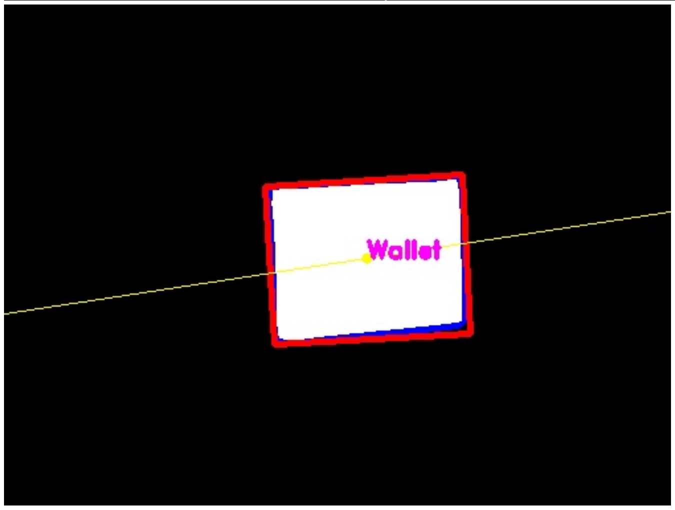




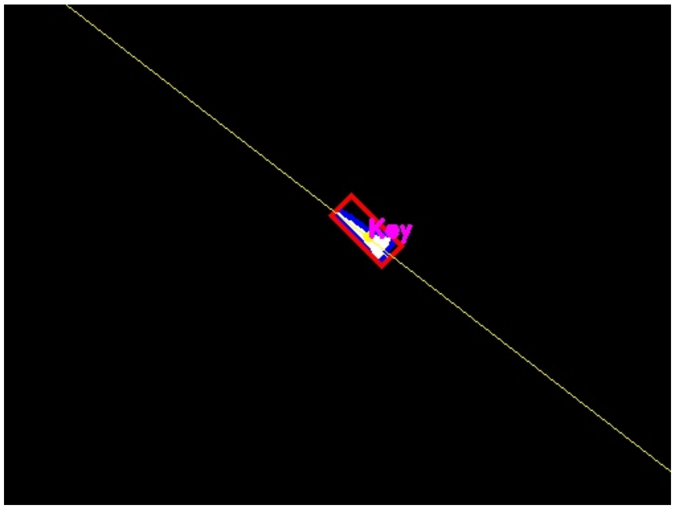


Object 9: Wallet



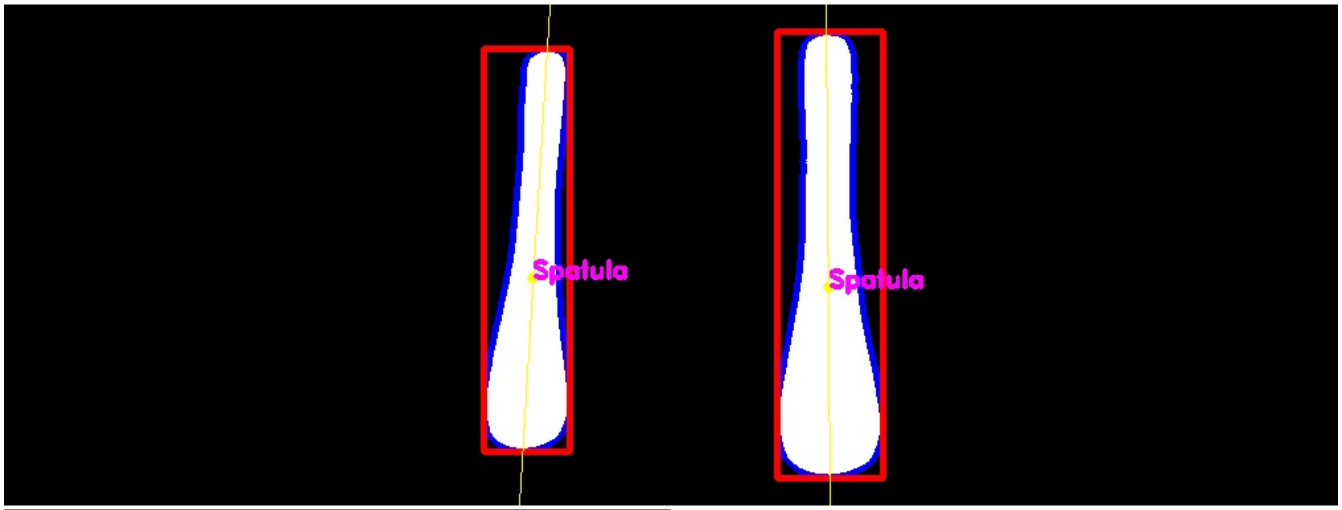


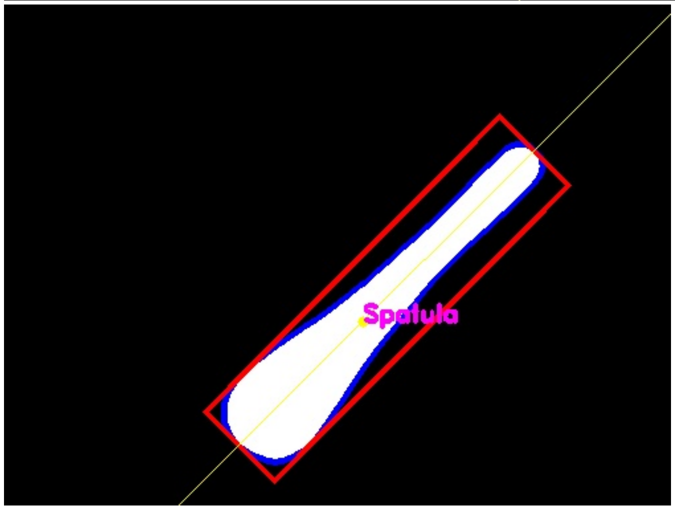
Object 10: Key



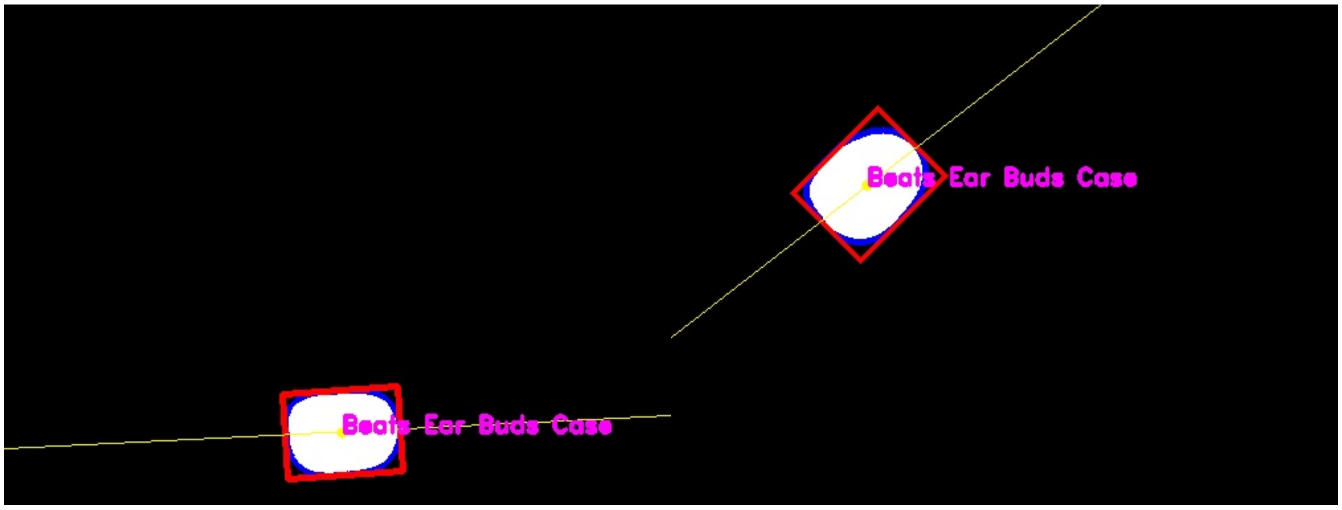
The key is also metallic but as a result of fine-tuning the thresholding values, I am able to classify it correctly.

Object 11: Spatula

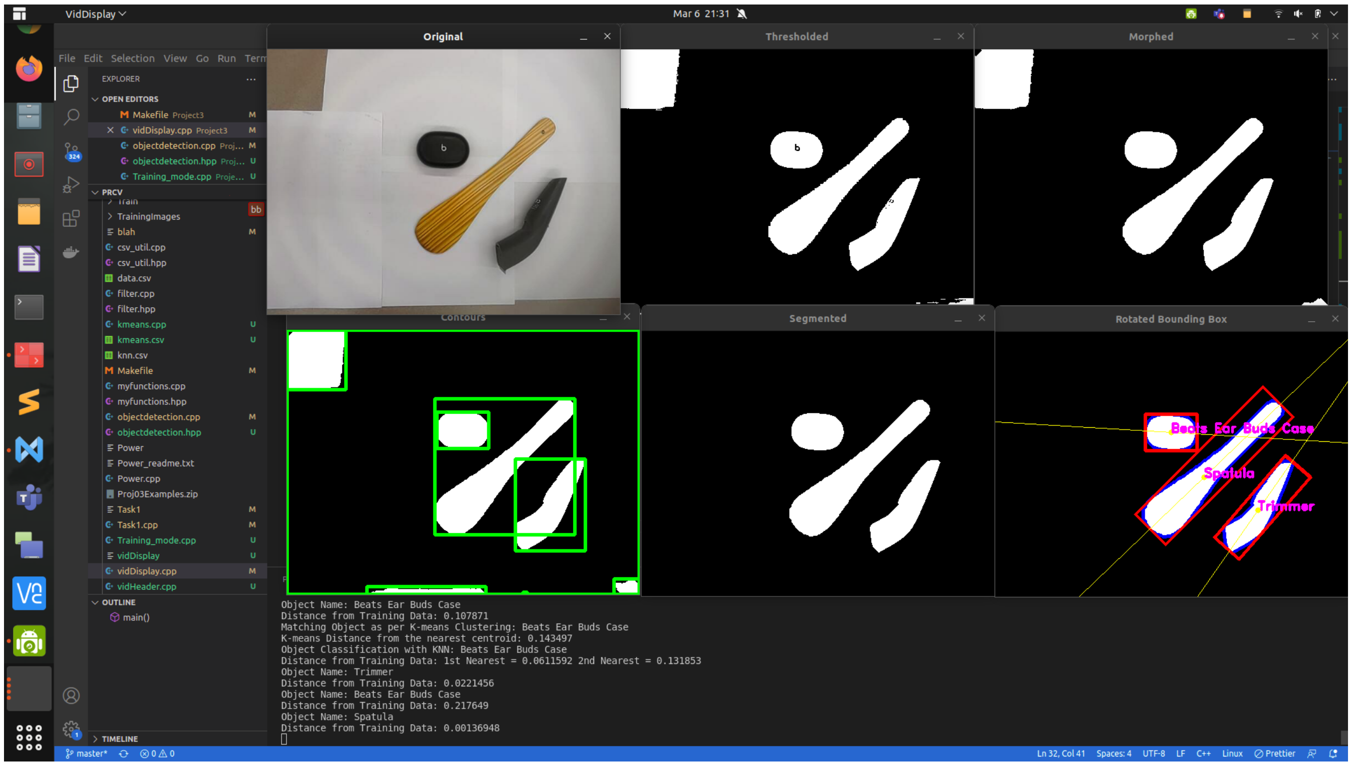




Object 12: Beats Ear Buds Case



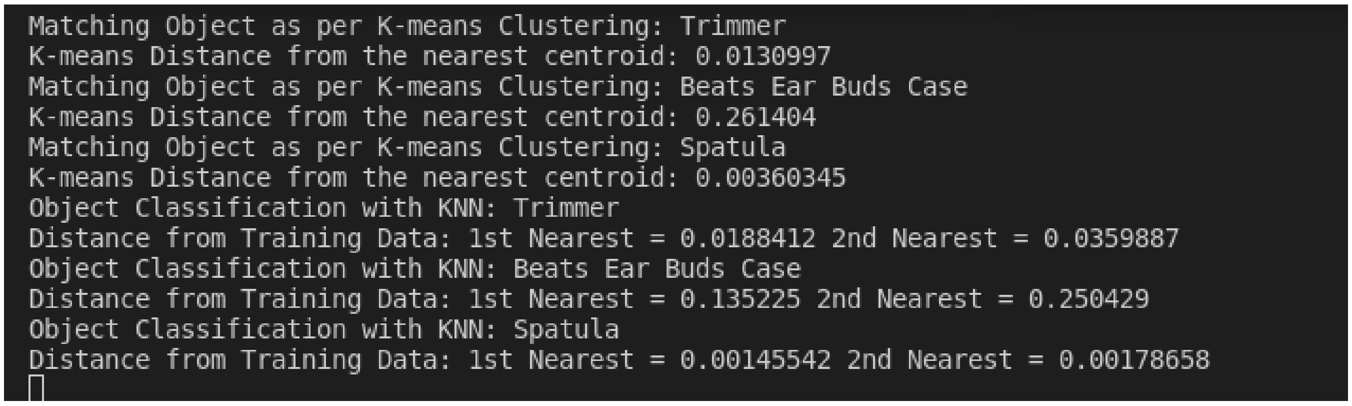
Multiple Objects Detection (Thoroughly elaborated in the Extension section):



```
Object Name: Beats Ear Buds Case
Distance from Training Data: 0.107871
Matching Object as per K-means Clustering: Beats Ear Buds Case
K-means Distance from the nearest centroid: 0.143497
Object Classification with KNN: Beats Ear Buds Case
Distance from Training Data: 1st Nearest = 0.0611592 2nd Nearest = 0.131853
Object Name: Trimmer
Distance from Training Data: 0.0221456
Object Name: Beats Ear Buds Case
Distance from Training Data: 0.217649
Object Name: Spatula
Distance from Training Data: 0.00136948
```

I have more information on this on extension but just wanted to put this here so as to get a comparison of how the model works for multiple objects. We can better see how the segmentation algorithm works as we have more than one region of interest. Also, it can be clearly seen through the terminal that all three objects are correctly classified through scaled Euclidean distance, through K-means and also through Knn.

Different Classifier Task 7: Explanation



```
Matching Object as per K-means Clustering: Trimmer
K-means Distance from the nearest centroid: 0.0130997
Matching Object as per K-means Clustering: Beats Ear Buds Case
K-means Distance from the nearest centroid: 0.261404
Matching Object as per K-means Clustering: Spatula
K-means Distance from the nearest centroid: 0.00360345
Object Classification with KNN: Trimmer
Distance from Training Data: 1st Nearest = 0.0188412 2nd Nearest = 0.0359887
Object Classification with KNN: Beats Ear Buds Case
Distance from Training Data: 1st Nearest = 0.135225 2nd Nearest = 0.250429
Object Classification with KNN: Spatula
Distance from Training Data: 1st Nearest = 0.00145542 2nd Nearest = 0.00178658
```

I have implemented both the Knn classification (k = 2) and the K-means clustering algorithm (k = 3) to classify and cluster a new object respectively. It is clear from the terminal output that the model is working correctly.

Evaluation of the Performance Task 8: Confusion Matrix

Confusion Matrix	**Beats Ear Buds Case**	**Spatula**	**Key**	**Wallet**	**Glass**	**Trimmer**	**Indian Candle Holder**	**Scissors**	**Spoon**	**Heart Candy**	**Hair Comb**	**Hair Comb Horizontal**
Beats Ear Buds Case	5	0	0	0	0	0	0	0	0	0	0	0
Spatula	0	5	0	0	0	0	0	0	0	0	0	0
Key	0	0	3	0	0	0	0	0	0	0	0	0
Wallet	1	0	0	4	0	0	0	0	0	0	0	0
Glass	0	0	0	0	5	0	0	0	0	0	0	0
Trimmer	0	0	0	0	0	5	0	0	0	0	0	0
Indian Candle Holder	0	0	0	0	0	0	4	0	1	0	0	0
Scissors	0	0	0	0	0	0	0	4	0	1	0	0
Spoon	0	0	0	0	0	0	0	0	5	0	0	0
Heart Candy	0	0	0	0	0	0	0	0	0	5	0	0

Hair Comb	0	0	0	0	0	0	0	0	0	1	4	0
Hair Comb Horizontal	0	0	0	0	0	0	0	0	0	0	0	5

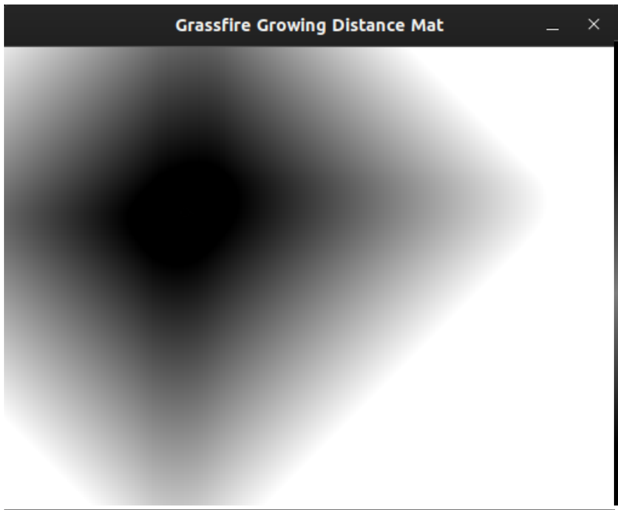
The above given table is a confusion matrix of 5 instances of object detection for each object. The table is copied directly from the .csv file. The value key has only 3 entries because the other 2 times, it wasn't classified at all. That is because there was so much white light from the reflection (the key is new and polished and from certain angles, the reflection was extremely high) that the region where the key was present was discarded as noise. For other objects, we can see that the model is fairly accurate even with using scaled euclidean distance as the distance metric. There are at most one incorrect classification out of 5. 7 objects out of 12 were classified with complete accuracy while 4 of the remaining had 1 incorrect classification and their distance from the said object was a close second.

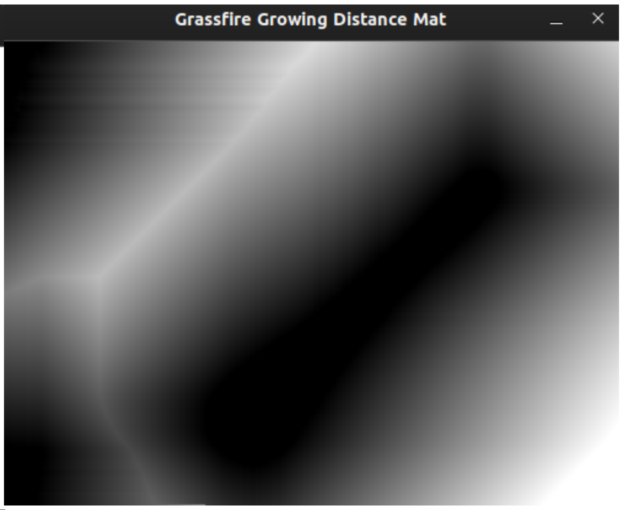
Link to the Demo Task 9:

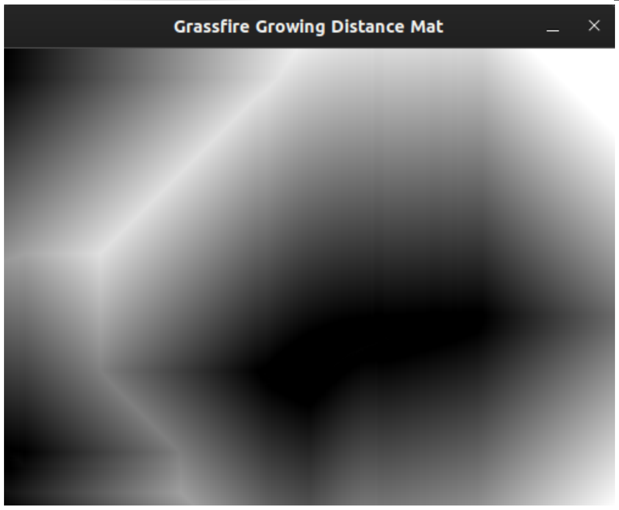
Following is the link to the recorded video:

https://drive.google.com/file/d/1QctGmDpDOcZlcuCX_Lyz_MohEMVd99At/view?usp=sharing

Extension 1: Grassfire Growing



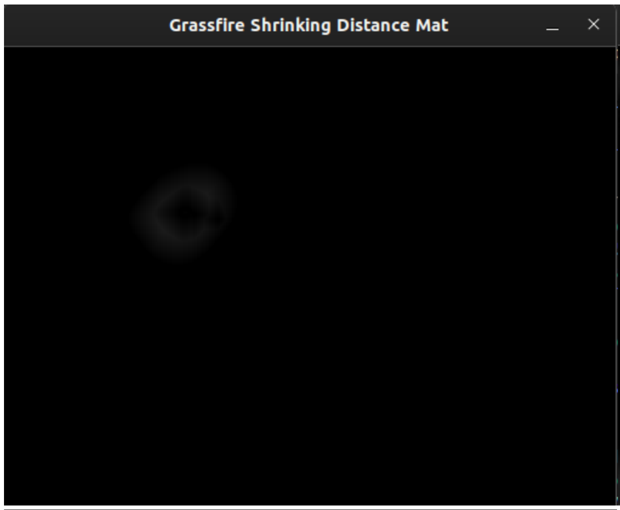


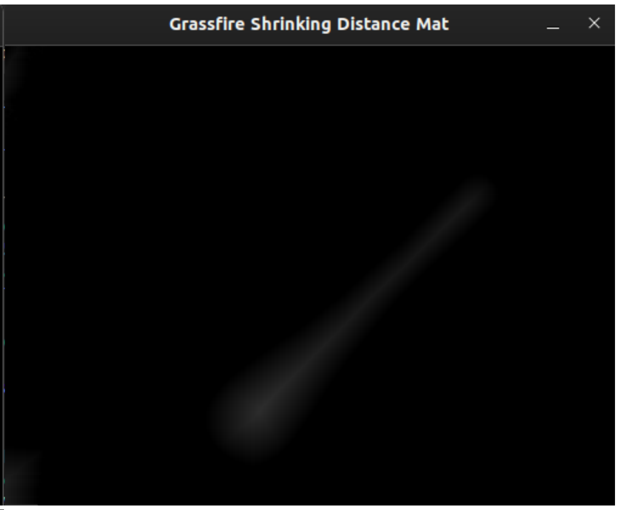


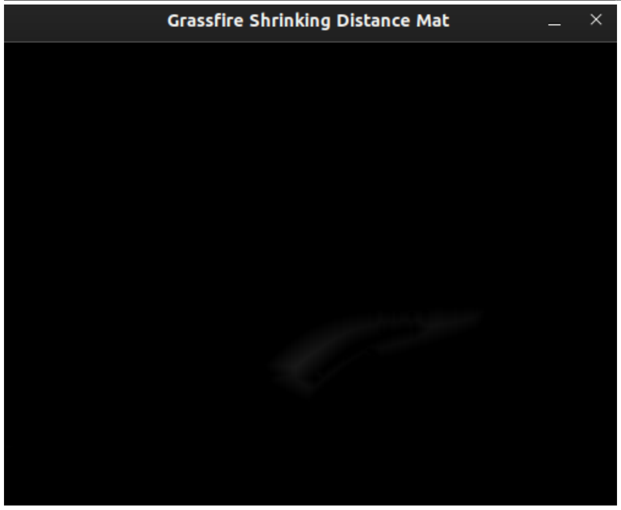
After completing the requirement of writing just one function from scratch (thresholding), I realized I wasn't completely satisfied with how the inbuilt algorithms and packages worked. So, I have implemented nearly all the functions myself with little or no help from the inbuilt functions. For implementing the grassfire growing algorithm, I first calculate the manhattan distance of the background from the foreground. I was getting errors initially but quickly realised it was because of using a not so sufficient datatype. unsigned char wasn't the best datatype as it could store values only till 255 and got reset after that. So, I implemented it using unsigned short datatype. The images above are of the matrix containing manhattan distance as the values. The images correspond to the three images that were shown for Task1-4.

The function also takes an argument count where you can specify the number of times you want to do growing.

Extension 2: Grassfire Shrinking







Similarly, I also implemented the grassfire shrinking algorithm from scratch. I just had to initialize the matrix in a different manner and calculate the manhattan distance of foreground from the background instead of the other way round. The above images show the distance matrix of the same 3 objects.

The function also takes an argument count where you can specify the number of times you want to shrink the image.

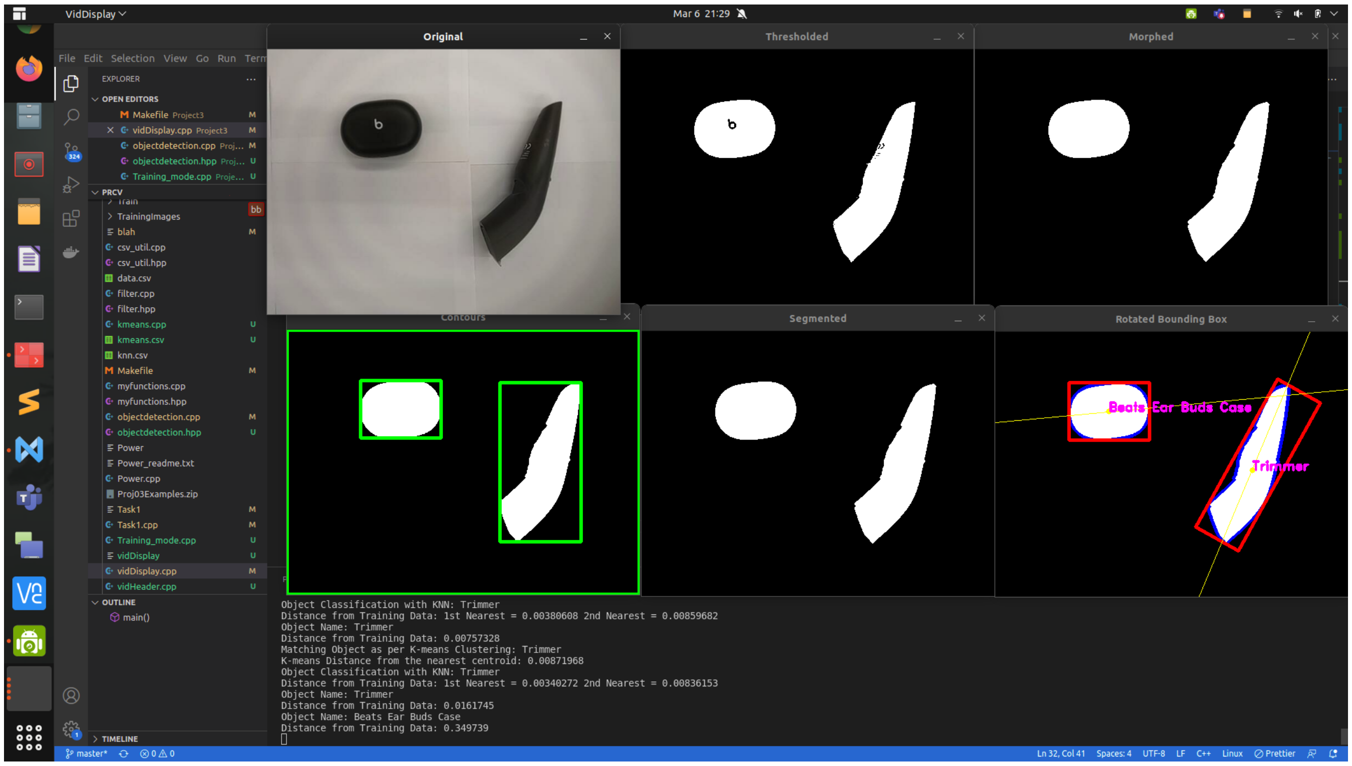
Extension 3: Segmentation

I initially just implemented connectedComponentWithStats function but soon realised it is not what I expected it to be. It did separate the regions but it left the smaller regions and had no functionality of removing the noise and providing me with the regions of interest specifically. Hence, I implemented my own segmentation algorithm using the stats and labels values that I got from the connectedComponentsWithStats function which allowed me to set an area threshold and also allowed me to remove the boundary noise which can be observed when comparing the images of task 2 and task 3.

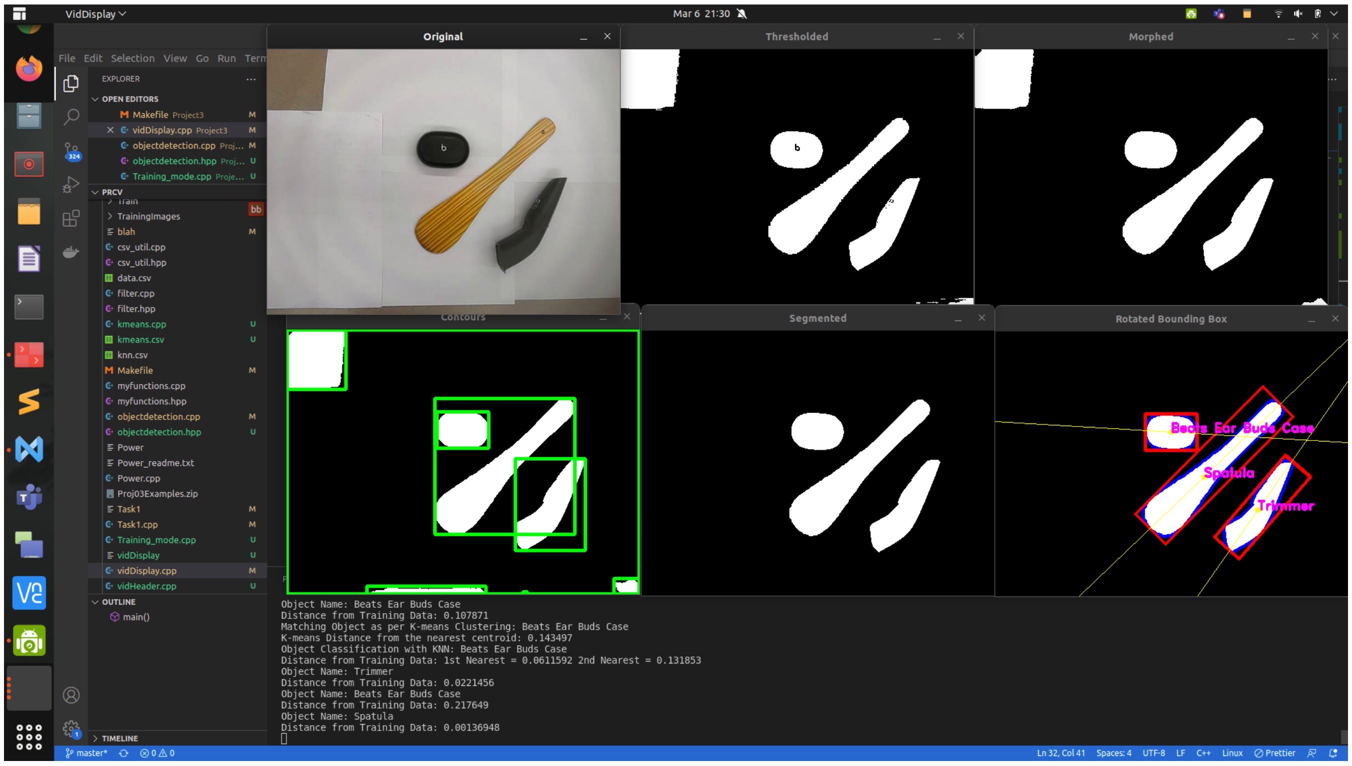
Extension 4: Adding more than 10 required objects

I have added other two more objects utilising the training mode of my model and they are classified correctly as can be seen from images in the classifying task.

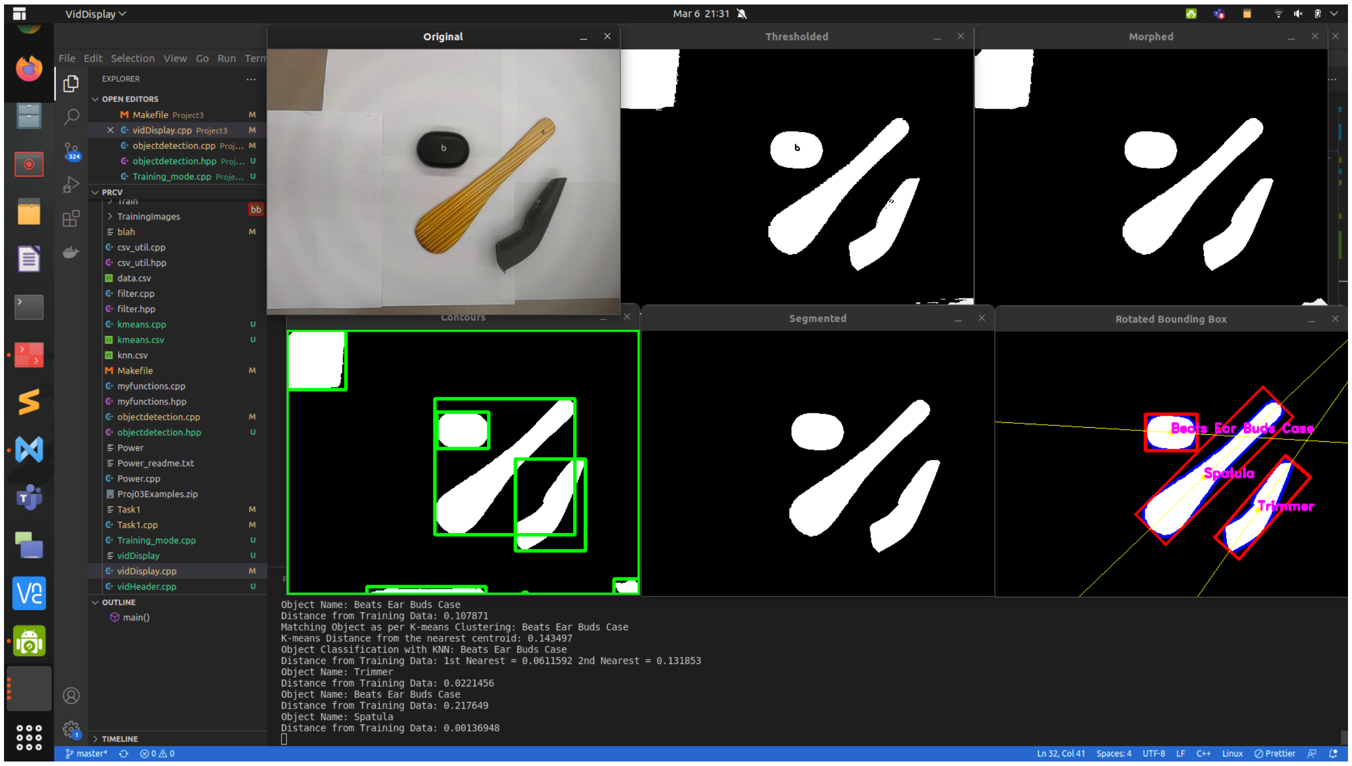
Extension 5: Multiple Object Recognition



```
Object Classification with KNN: Trimmer
Distance from Training Data: 1st Nearest = 0.00380608 2nd Nearest = 0.00859682
Object Name: Trimmer
Distance from Training Data: 0.00757328
Matching Object as per K-means Clustering: Trimmer
K-means Distance from the nearest centroid: 0.00871968
Object Classification with KNN: Trimmer
Distance from Training Data: 1st Nearest = 0.00340272 2nd Nearest = 0.00836153
Object Name: Trimmer
Distance from Training Data: 0.0161745
Object Name: Beats Ear Buds Case
Distance from Training Data: 0.349739
```

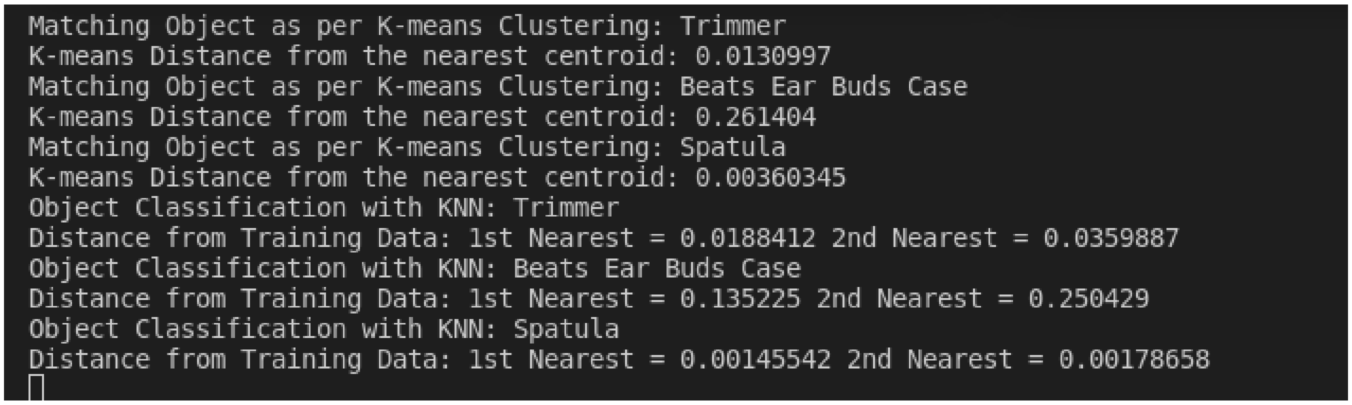


```
Object Name: Beats Ear Buds Case
Distance from Training Data: 0.107871
Matching Object as per K-means Clustering: Beats Ear Buds Case
K-means Distance from the nearest centroid: 0.143497
Object Classification with KNN: Beats Ear Buds Case
Distance from Training Data: 1st Nearest = 0.0611592 2nd Nearest = 0.131853
Object Name: Trimmer
Distance from Training Data: 0.0221456
Object Name: Beats Ear Buds Case
Distance from Training Data: 0.217649
Object Name: Spatula
Distance from Training Data: 0.00136948
```



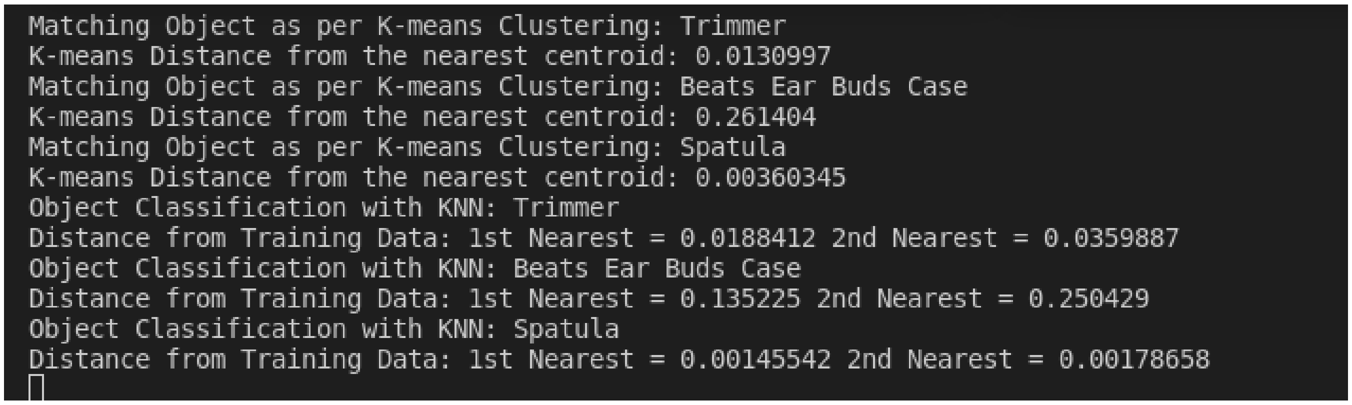
```
Object Name: Beats Ear Buds Case
Distance from Training Data: 0.107871
Matching Object as per K-means Clustering: Beats Ear Buds Case
K-means Distance from the nearest centroid: 0.143497
Object Classification with KNN: Beats Ear Buds Case
Distance from Training Data: 1st Nearest = 0.0611592 2nd Nearest = 0.131853
Object Name: Trimmer
Distance from Training Data: 0.0221456
Object Name: Beats Ear Buds Case
Distance from Training Data: 0.217649
Object Name: Spatula
Distance from Training Data: 0.00136948
```

My model can work without a single change in the code and without glitching to classify a single object or multiple objects at once. To provide my code with this flexibility, every feature and other components that I calculate as a part of my code has an outer vector component which basically is of the size of the number of regions of interest identified which is one for single object detection and more if the feed detects more than one objects at once. All my functions are designed to accommodate multiple objects at once in the training as well as the prediction and classification part, even the knn and the k-means algorithms. As can be seen from the terminal output above and below that all the objects are detected and classified successfully using all three of my algorithms, scaled euclidean distance, knn and k-means as well. This is something that I was extremely happy to see and it worked on the first try!!



```
Matching Object as per K-means Clustering: Trimmer
K-means Distance from the nearest centroid: 0.0130997
Matching Object as per K-means Clustering: Beats Ear Buds Case
K-means Distance from the nearest centroid: 0.261404
Matching Object as per K-means Clustering: Spatula
K-means Distance from the nearest centroid: 0.00360345
Object Classification with KNN: Trimmer
Distance from Training Data: 1st Nearest = 0.0188412 2nd Nearest = 0.0359887
Object Classification with KNN: Beats Ear Buds Case
Distance from Training Data: 1st Nearest = 0.135225 2nd Nearest = 0.250429
Object Classification with KNN: Spatula
Distance from Training Data: 1st Nearest = 0.00145542 2nd Nearest = 0.00178658
```

Extension 6: Implementation of K-means clustering algorithm from scratch to cluster the new object into trained clusters


```
Matching Object as per K-means Clustering: Trimmer
K-means Distance from the nearest centroid: 0.0130997
Matching Object as per K-means Clustering: Beats Ear Buds Case
K-means Distance from the nearest centroid: 0.261404
Matching Object as per K-means Clustering: Spatula
K-means Distance from the nearest centroid: 0.00360345
Object Classification with KNN: Trimmer
Distance from Training Data: 1st Nearest = 0.0188412 2nd Nearest = 0.0359887
Object Classification with KNN: Beats Ear Buds Case
Distance from Training Data: 1st Nearest = 0.135225 2nd Nearest = 0.250429
Object Classification with KNN: Spatula
Distance from Training Data: 1st Nearest = 0.00145542 2nd Nearest = 0.00178658
```

I understand that k-means is a clustering algorithm and not exactly a classification algorithm but the train of thought I had was that I would cluster the training data where k equals the number of objects and calculate the centroids accordingly. Now, when I detect one or more objects on my feed, I would measure their distance from each centroid and find the centroid which has the minimum distance from the feature vector/s of the object/s to be classified. Then, I would put the objects into the cluster whose centroid has the minimum distance from the feature vector of the object and the label corresponding to that centroid should be the classification of my object as well. I just started this with experimentation but it worked really well as can be seen from the above images and also worked extremely well for all the three objects of the database put all at once as well as when putting one by one and hence I felt like it would make an excellent extension.

Reflection:

This project has so far been the best learning outcome of this course and it has me excited for all the future projects that I will get to do. I am learning so much about the language C++, various data structures in C++, manipulation and the OpenCV package. I have lost track of how many hours I spent on the official documentation of openCV and on google learning how to use vectors in C++, about iterating through a for loop in different ways, about pointers and about file handling and programming techniques and organisation in general. At every step of this project, I realised that I wasn't completely satisfied with the way inbuilt functions worked and have either discarded them completely or built on them to write my own functions which would work the way I want them to. This would have been a daunting task just two months back but for this project, I didn't even think twice about writing them. I am so much grateful to Professor Maxwell and want to thank him for his pedagogy as I am able to implement functions by just following his instructions and in-depth explanation of concepts. From implementing grassfire growing and shrinking to my own segmentation code, constructing oriented bounding boxes, computing angle of the central axis and all the features invariant to translation, rotation and scale and to implementing extensions, I have experienced phenomenal growth in my understanding of the course and as a programmer as well. There were days I got so intrigued I could not even get away from my laptop. From not having used C++ at all just two months back to implementing the whole k-means clustering algorithm from scratch in the red-eye (university shuttle to facilitate students' ride home) which takes about 25 mins, I can see the growth I have achieved under the wing of Professor Maxwell. I am looking forward to giving my best in the upcoming projects.

Acknowledgement:

Official documentation of OpenCV has been exhaustively referred for the completion of this project. I would like to thank Professor Bruce Maxwell for his easy to follow instructions and in-depth explanation of concepts which enabled me to bring this project to fruition. I would also like to thank my classmates and friends Ravina Lad and Sumegha Singhania for the various brainstorming sessions and the constant support they have provided me with. I have also googled a lot of stuff as and when required without going through actual websites that I can put as a reference but StackOverflow and geeksforgeeks definitely are worth a mention.