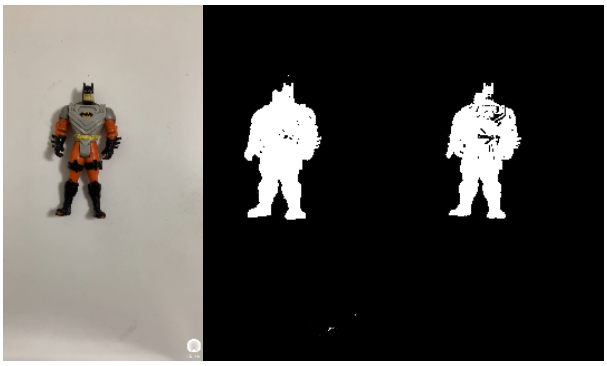
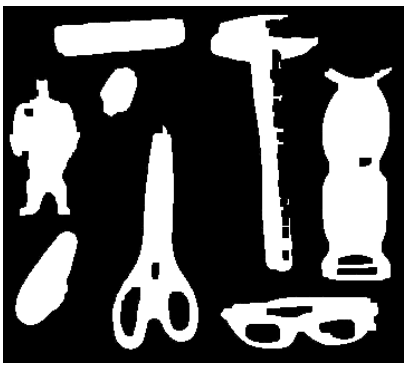
Project 3: Real-time Object 2-D Recognition

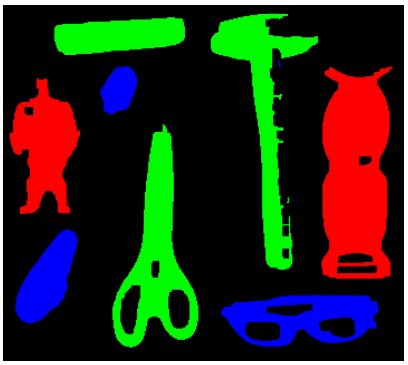
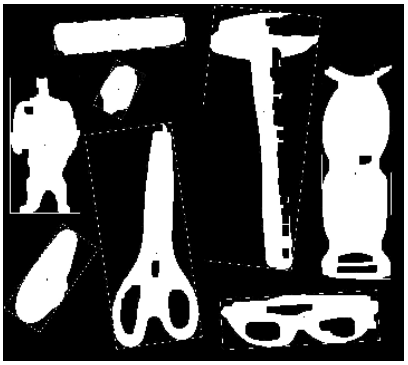
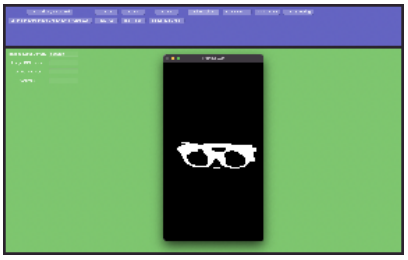
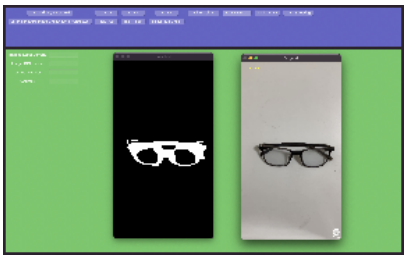
The assignment stated with identifying a tool that could make me use the phone or ipad as the main camera for live streaming and finding a white background surface, finding the perfect light and balancing the ipad between books at the right height.

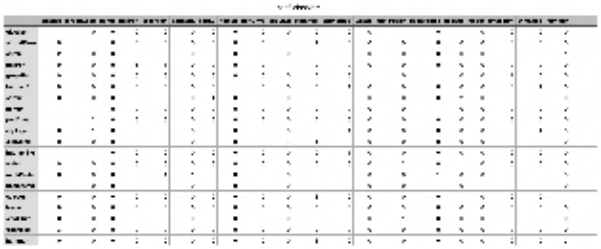
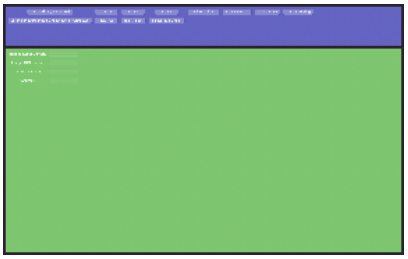
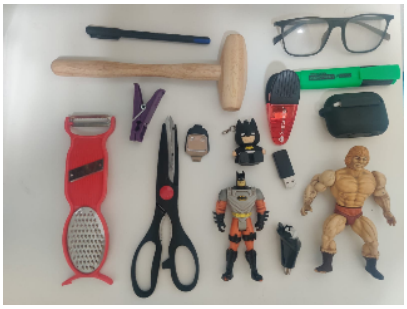
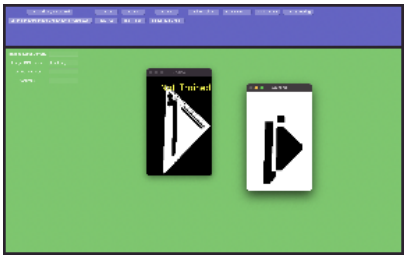

The next task was to identify the objects that can be used to perform object detection. Few of the objects are shown above.

This project deals with 2D object recognition when objects are kept in a white background, the features are selected such that they are translation, scale, rotation invariant and they are normalized so that everything falls in almost the same range. Developed a GUI, method to train for new objects, store the images to pull more new features, and train it better. Live object detection is performed.

Task	Button	Comments	Result

Threshold the input Video	Task 1	Implemented three methods of thresholding. • Created my own implementation of binary image, where any pixel having 200-250 range values in R or G or B channel is made to black and other pixels are made to white. • Converted the image to HSV and removed the pixels falling below 50 saturation. • Converted image to greyscale and thresholded on it using the in-built function calls. • Used OTSU thresholding to differentiate the background and the foreground clearly for better evaluation due to the setup issues.	Threshold Comparisons   
Clean up the binary image	Task 2	To cover the missing spots in the object not detected during thresholding, I use binary image cleanup. Used 2 erosion to remove the small noises and then Growing(Dilation) 12 times followed by 10 Shrinking(erosion) operations. This improved the coverage between the body but the outlines got disturbed when the distance between the boundary is close like the leg between the batman. As we have a similar total number of shrink and grow, we get back the actual shape of the object even though there might be boundary issues	

Segment the image into regions	Task 3	Identified the different regions with ConnectedComponentswithStats to identify the regions and then we remove the objects below a certain threshold so that unknown random shadow detections can be removed. We use three colors to color the detected regions alternatively.	
Compute features for each major region	Task 4	From the given frame we identify the objects and then draw the oriental bounding box and the centroid for each individual region. To draw the oriental bounding box, I have used the minRectangle function, passing all the coordinates of each individual region. Centroid is plotted using the stats returned by the ConnectedComponentswithStats. Humoments feature is generated and returned to be stored in the training data. Applied log normalization for all the humoment feature vector.	
Collect training data	Task 5	The label on the GUI needs to be provided with the object label that is going to be trained. And once the button is clicked it opens up a frame, when the used presses 'n' then the original image is stored and the humoments feature vector is stored in the csv along with the label name.	
Classify new images	Task 6	When the frame is showed if the user presses 'd' then detection is called and then either one of the Euclidean Distance, Manhattan Distance or chiSquare distance is used to identify the closest vector to the current object and the a text is created on the output image showing the detected label.	
Implement a different classifier	Task 7	Implemented KNN classifier with K = 10, and same as above the feature vectors are generated and compared. Out of the top 10 closest we find the max labels and print that on the output image screen.	

Evaluate the performance of your system	Task 8	From a collection of test data where 3 images per object is used to check the accuracy. Computed accuracy with manhattan, euclidean, chisquare distances. And created a confusion matrix for the euclidean distance which performed the best among them.	Euclidean Distance Accuracy = 20 Manhattan Distance Accuracy = 13 chiSquare Distance Accuracy = 3.33 The low accuracy can be due to multiple reasons, mainly lot of items have almost similar shape, if we consider the number of pixels then we can improve the accuracy, and if we consider the color of objects as feature then it can improve more. 
Capture a demo of your system working	Task 9	Video shows how various operations are to be performed from task 1 to task 9 and the extensions and the live detections.	Video of the working system
GUI	Extension 1	Developed GUI, can make it better by showing the imshow results on the GUI instead of a popup.	
More Objects	Extension 2	Object Detection was implemented for 20 objects.	
Known or Unknown Object Detection	Extension 3	Kept a threshold of 2 units and if none of the feature vector euclidean distance falls under that then we label it as unknown object.	
Experimenting with distance metrics	Extension 4	Compared results with 3 different distance metric Euclidean, Manhattan and chiSquare.	Explained above in task 8

Key Takeaways:

Understood how humoments work for object classification, and how k-nearest neighbour algorithm works. Understood the methods and the reasons for why thresholding fails in certain scenarios and why some thresholding method is better than others. Morphological operations were tested and understood reasons why sometimes its better to shrink before starting as it will remove small noises. Got a deeper understanding of computer vision.

Acknowledgment

Following websites were referred to learn about 2D object detection:

- https://opencv.org
- https://stackoverflow.com/
- https://learnopencv.com/shape-matching-using-hu-moments-c-python/
- https://answers.opencv.org/question/74482/what-does-humoments-tell-me/
- wxgui : Lot of resources, StackOverflow but the best lib was: https://github.com/gammasoft71/Examples_wxWidgets/blob/master/src/CommonControls/Button/Button.cpp