Project 3: Real-time Object 2-D recognition

Name: Suparna Srinivasan

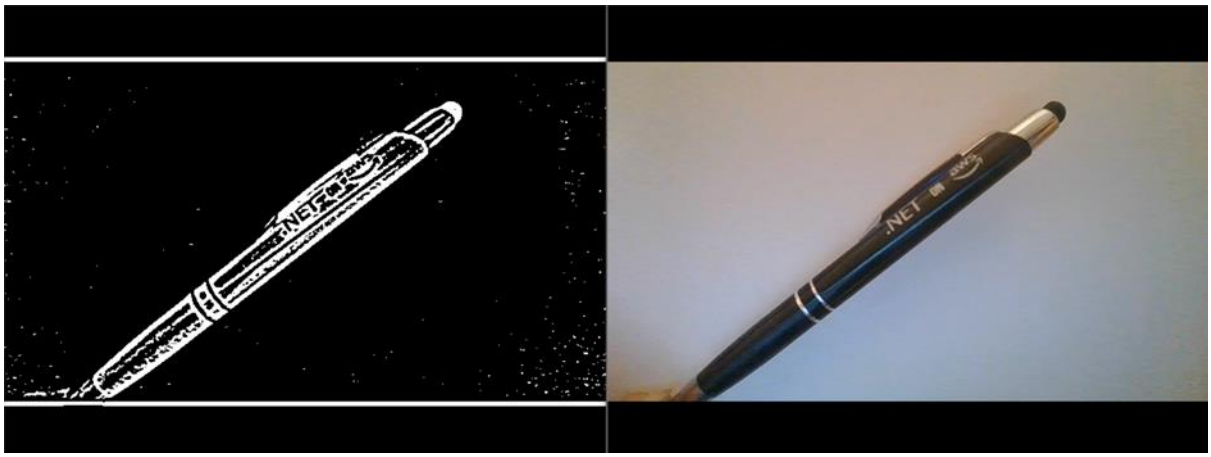
This project was about 2d object recognition. The goal was to have the computer's webcam identify a specified set of objects placed on a white surface in a translation, scale, and rotation-invariant manner from a camera looking straight down. In real time, the camera captures a video and applies thresholding, cleaning, segmentation, region classification, and collects training data from the camera on the user's key press.

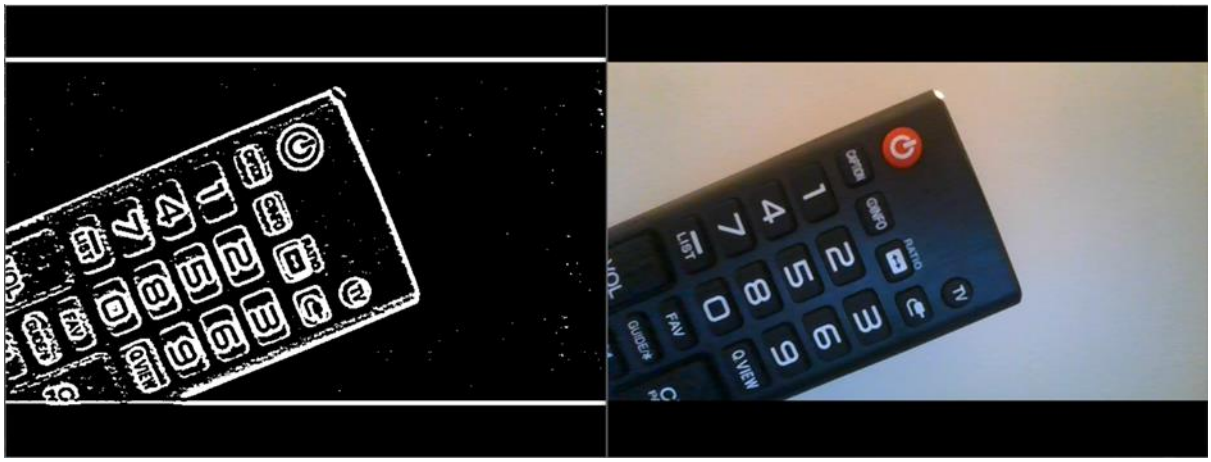
Tasks

Task 1: Threshold the input video

I have started building OR system by calling CV's thresholding algorithm that separates an object from the background. It displays the thresholded video. I have taken images where objects are darker than their background.

I have blurred the images a little so they can be easily recognized (fork, pen, remote).





The left is the thresholded image and the right is the original, all taken from the laptop's live video camera. You can see a lot of noise and holes in the thresholded images.

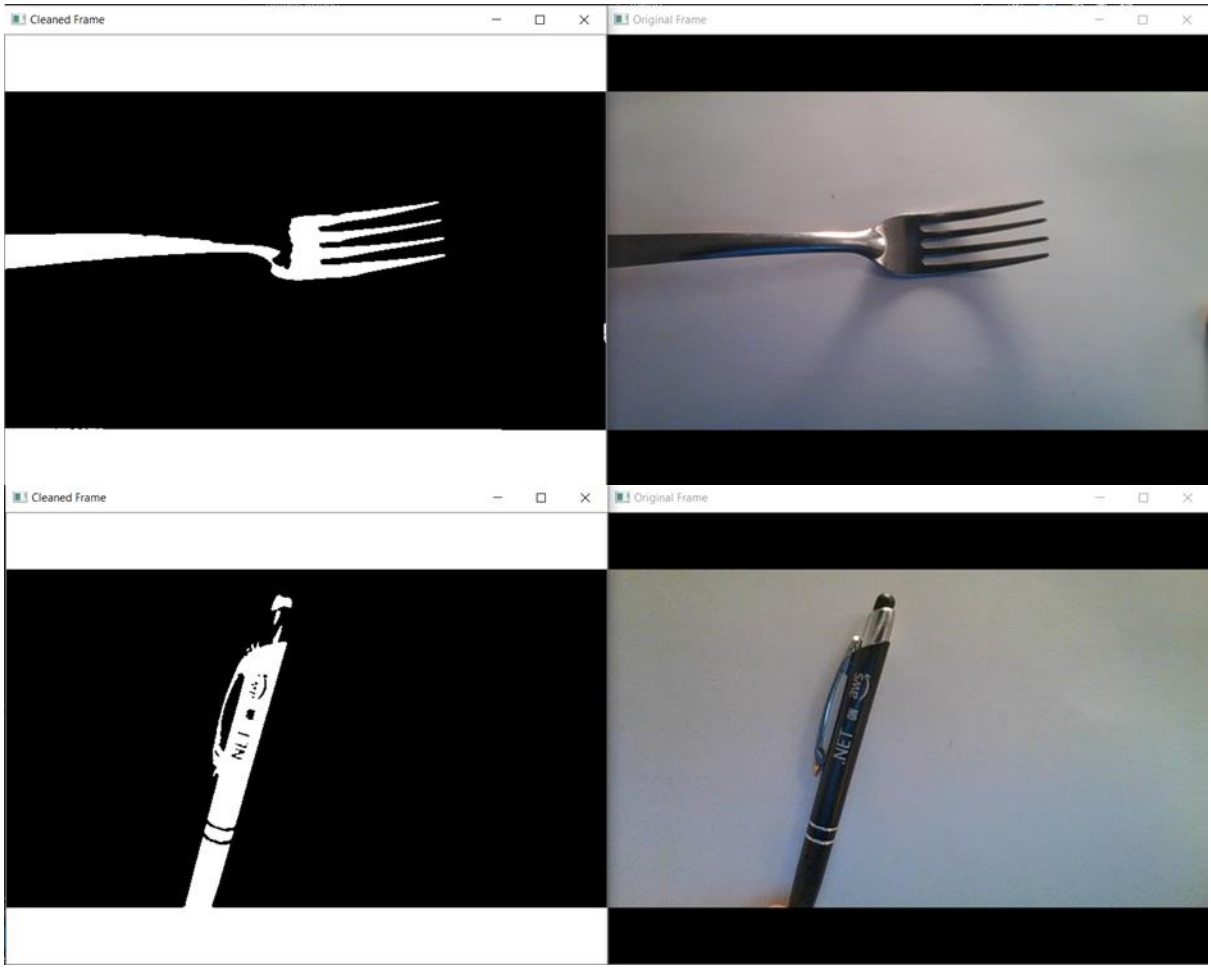
Task 2: Clean up the binary image

I cleaned thresholded images using morphological filtering. Morphological filtering was implemented from scratch and not used CV's inbuilt functions. The `morph_cleanup` function takes an input image, which is the binary image obtained after thresholding the distance transformed image. The function then applies either dilation or erosion to the image based on the specified operation, which is passed as a string argument.

For dilation, the function checks the pixel values in the input image, and if the pixel value is less than a threshold value (15 in this case), it sets the corresponding pixel value in the output image to 255 (white). Otherwise, it sets the output pixel value to 0 (black). This has the effect of expanding the bright regions in the input image.

For erosion, the function performs a similar operation, but in reverse. It sets the output pixel value to 0 (black) if the corresponding pixel value in the input image is less than the threshold value, and sets it to 255 (white) otherwise. This has the effect of shrinking the bright regions in the input image.

After applying the morphological operation, the function produces an output image with the filtered result.

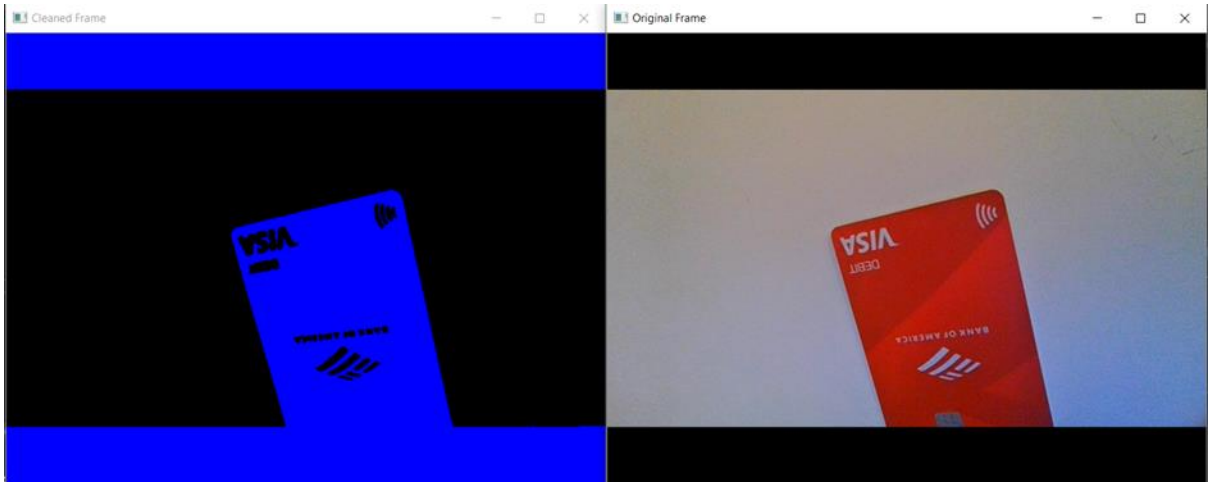


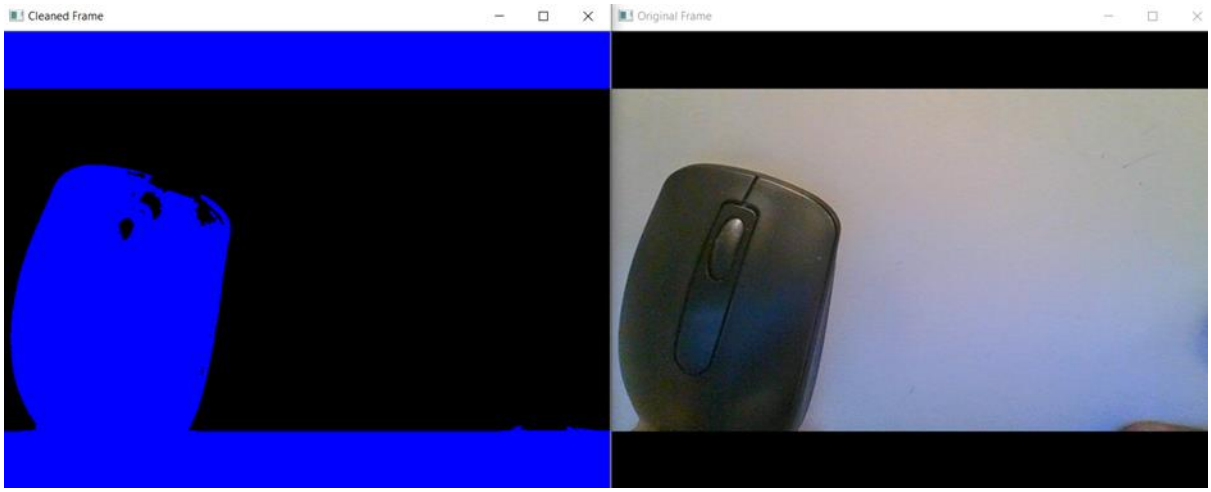
Task 3: Segment the image into regions

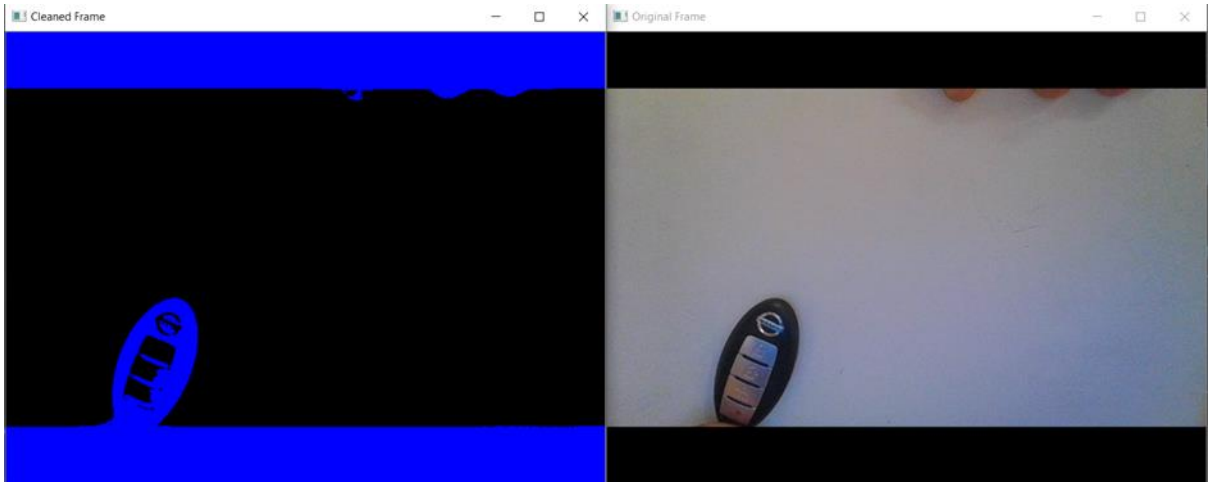
The filtered image is then used for further processing, such as segmentation into different regions, and is achieved by applying the `cv::connectedComponentsWithStats()` function to the preprocessed input image. This function assigns each pixel in the image to a connected component based on its pixel value and connectivity and returns a labeled image where each component is assigned a unique integer label.

To visualize the components, a random color map is created and used to assign a color to each component in the labeled image. The color map is defined such that the background is black, and each component is assigned a unique color (in this case, blue). Components that do not meet the minimum size requirement (defined as 1000 pixels in this case) are not displayed.

I then compute the list of region IDs sorted by area, where each region corresponds to a connected component. The regions are sorted in descending order of area, which allows the code to select the largest regions for further processing.



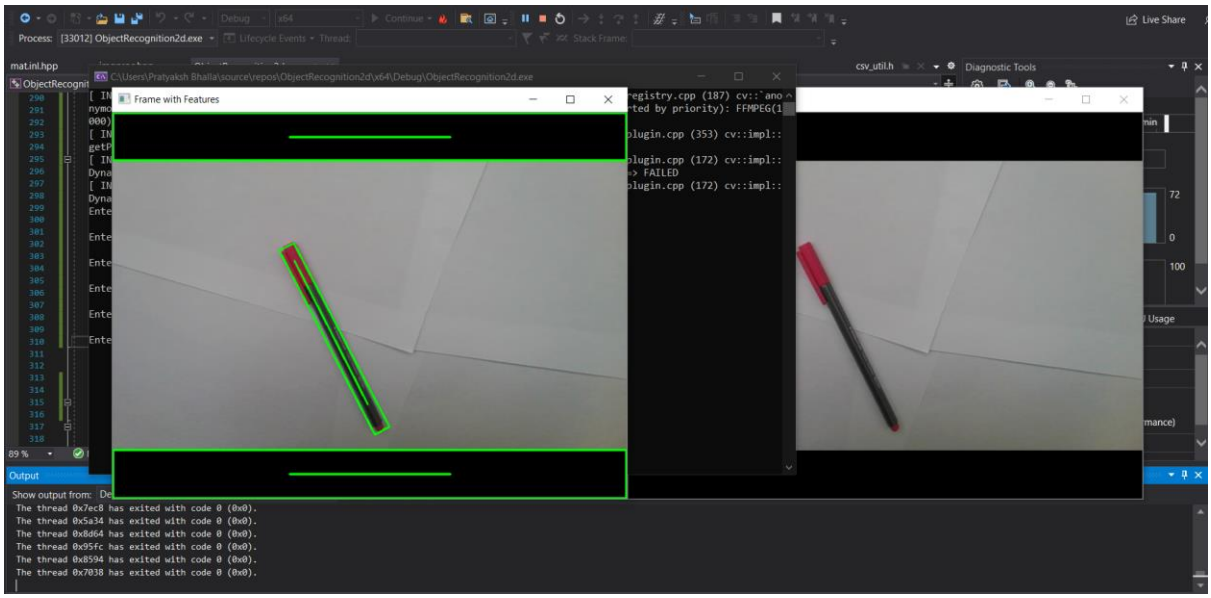


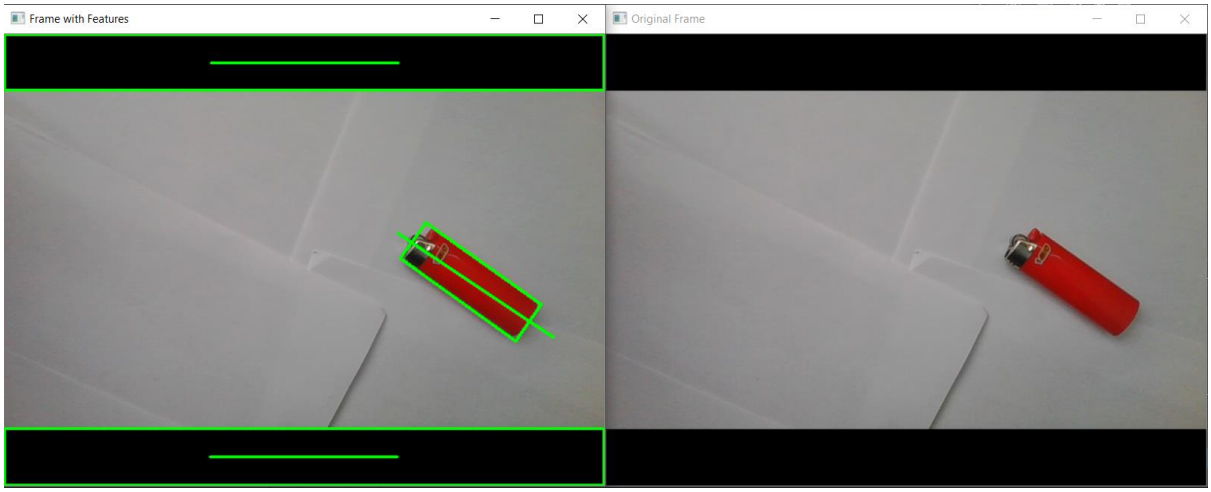


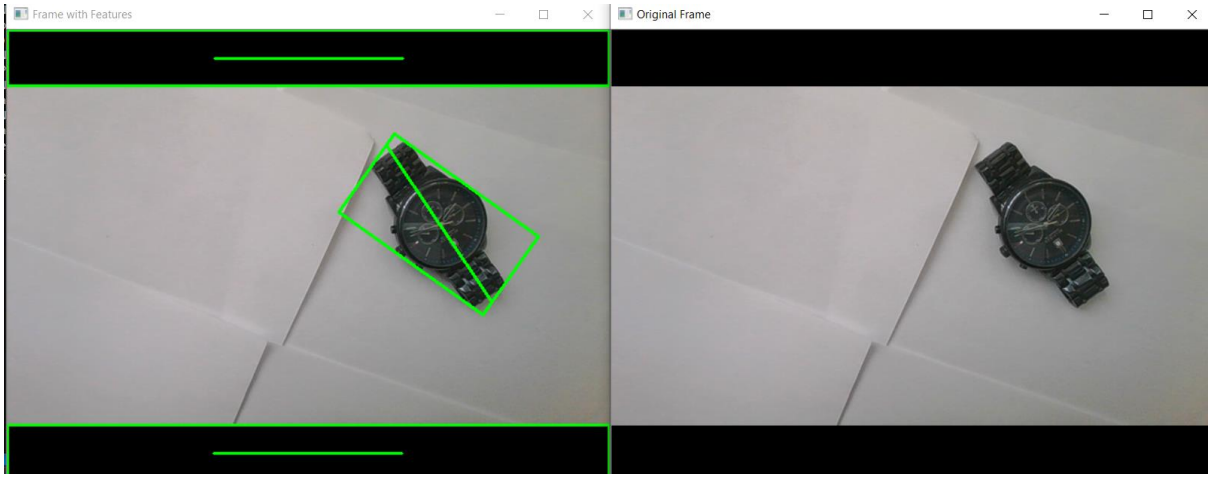
Task 4: Compute features for each major region

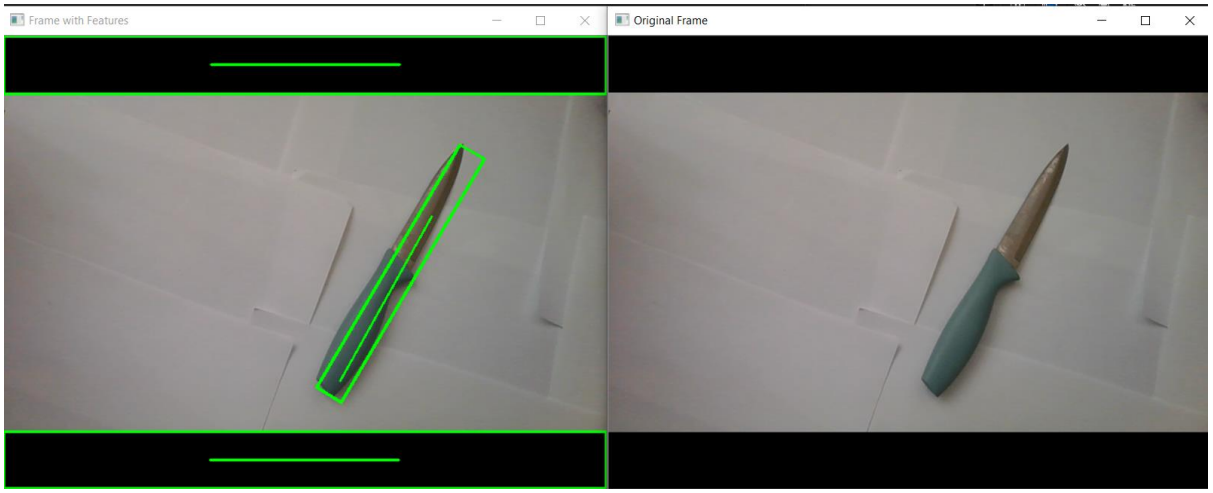
Features are computed for each of the largest N regions in the preprocessed input image, where N is set to 3. The code first computes a mask for each region using the `cv::connectedComponentsWithStats()` function, and then calculates the moments and oriented bounding box of each region using the `cv::moments()` and `cv::minAreaRect()` functions. The resulting features, including the x and y centroids, orientation angle, and width and height of the bounding box, are then added to a feature vector for each region. The

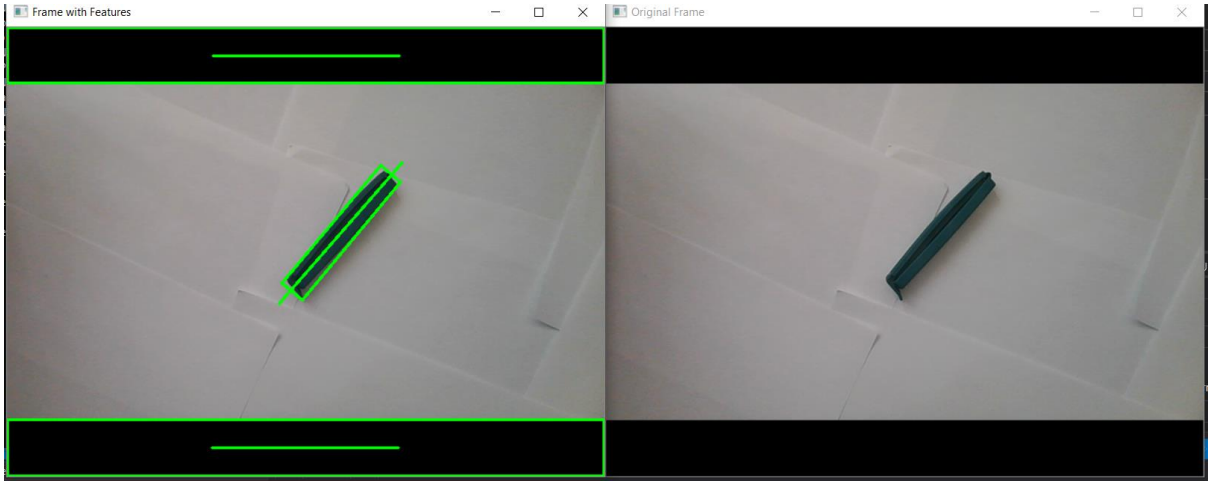
feature vectors for all regions are stored in a vector, and the feature vector for the last processed region is stored separately.

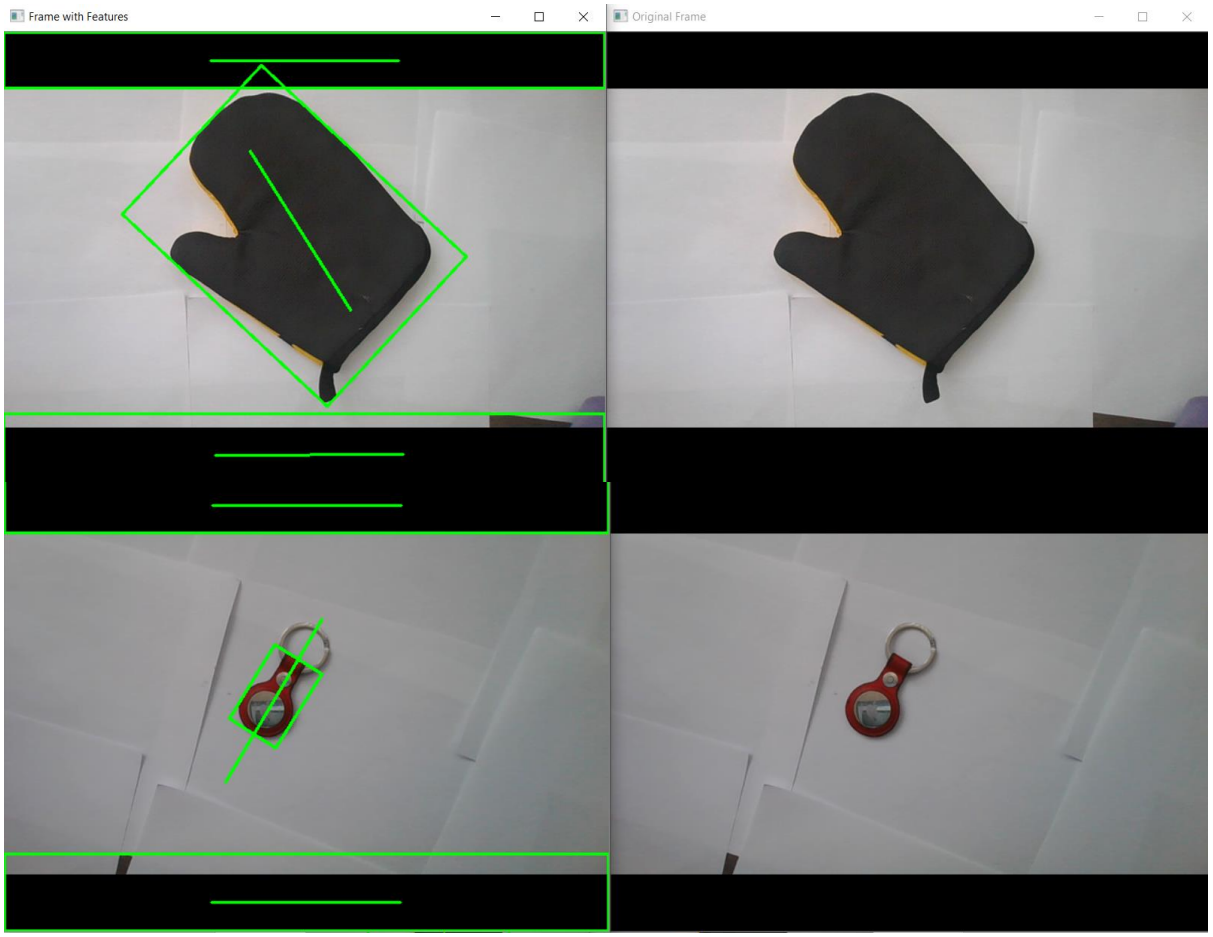


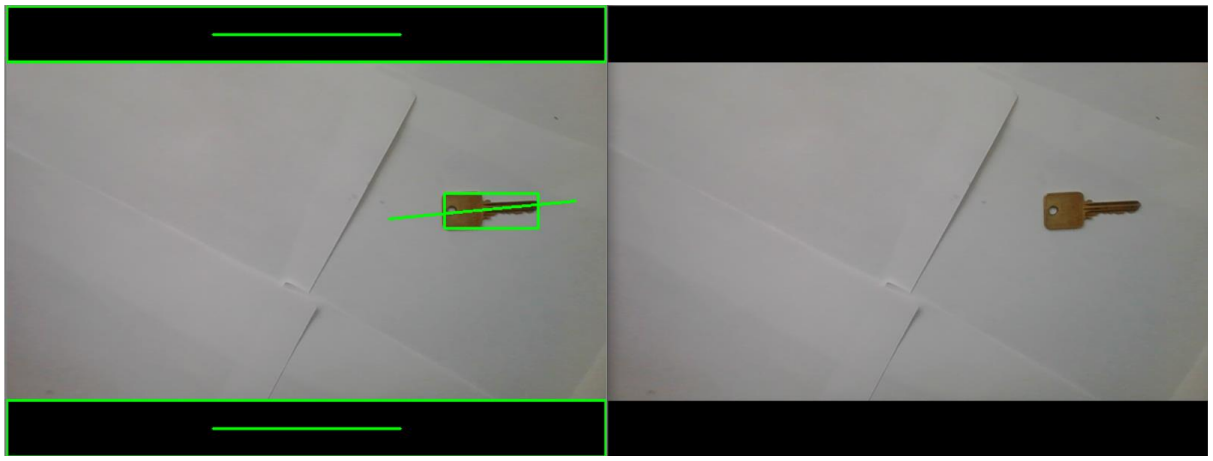


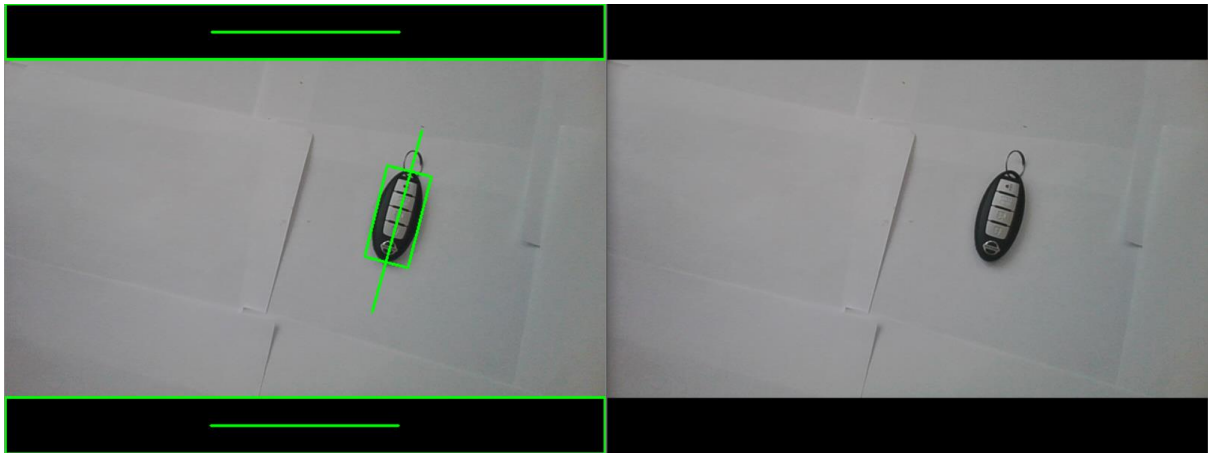



Frame with Features
Original Frame


Frame with Features
Original Frame


Frame with Features
Original Frame

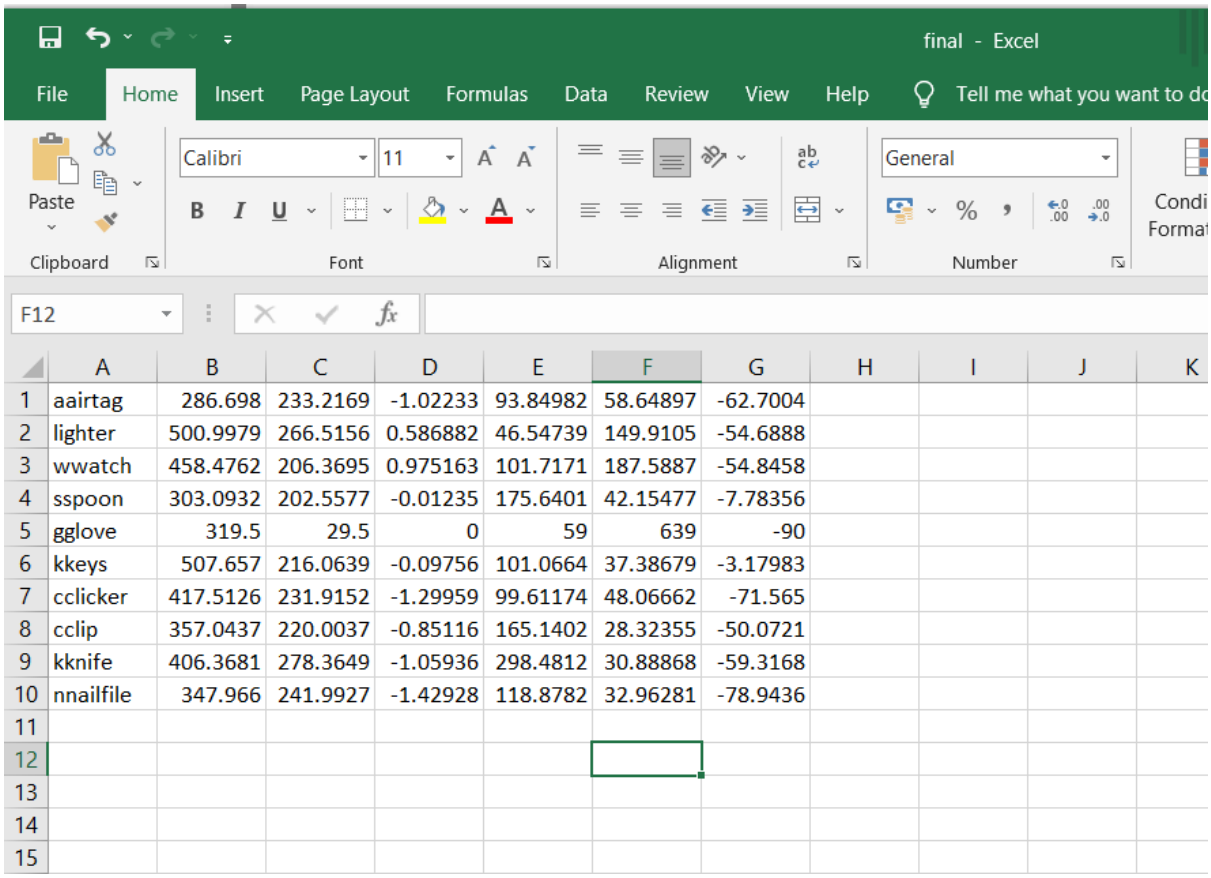




Task 5: Collect training data

On key press ‘n’ The user is prompted to enter the name of the object being trained, and the program concatenates the entered name with the path "data/" and ".csv" to create a filename for the corresponding CSV file. The function `append_image_data_csv()` is then called to append the feature vector of the last processed region to the specified CSV file.

The final csv comprises of all 10 objects and their feature vectors.



final - Excel

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do

Paste Clipboard Calibri 11 Font Alignment General Number Condi Format

F12

	A	B	C	D	E	F	G	H	I	J	K
1	aairtag	286.698	233.2169	-1.02233	93.84982	58.64897	-62.7004				
2	lighter	500.9979	266.5156	0.586882	46.54739	149.9105	-54.6888				
3	wwatch	458.4762	206.3695	0.975163	101.7171	187.5887	-54.8458				
4	sspoon	303.0932	202.5577	-0.01235	175.6401	42.15477	-7.78356				
5	gglove	319.5	29.5	0	59	639	-90				
6	kkeys	507.657	216.0639	-0.09756	101.0664	37.38679	-3.17983				
7	cclicker	417.5126	231.9152	-1.29959	99.61174	48.06662	-71.565				
8	cclip	357.0437	220.0037	-0.85116	165.1402	28.32355	-50.0721				
9	kknife	406.3681	278.3649	-1.05936	298.4812	30.88868	-59.3168				
10	nnailfile	347.966	241.9927	-1.42928	118.8782	32.96281	-78.9436				
11											
12											
13											
14											
15											

Reflection:

It was a really fun and challenging project. Although I was not able to complete tasks 6-9, the first 5 tasks were completed successfully. I had trouble setting up the white background and taking a picture of the objects, making sure there were no shadows, etc. I learned a lot about images, their properties such as thresholding, cleaning, morphological filtering, etc and recognizing them in real time.