Syed Azhar Hussain Quadri

CS 5330 PRCV

Project 3 Report

Real-time 2-D Object Recognition

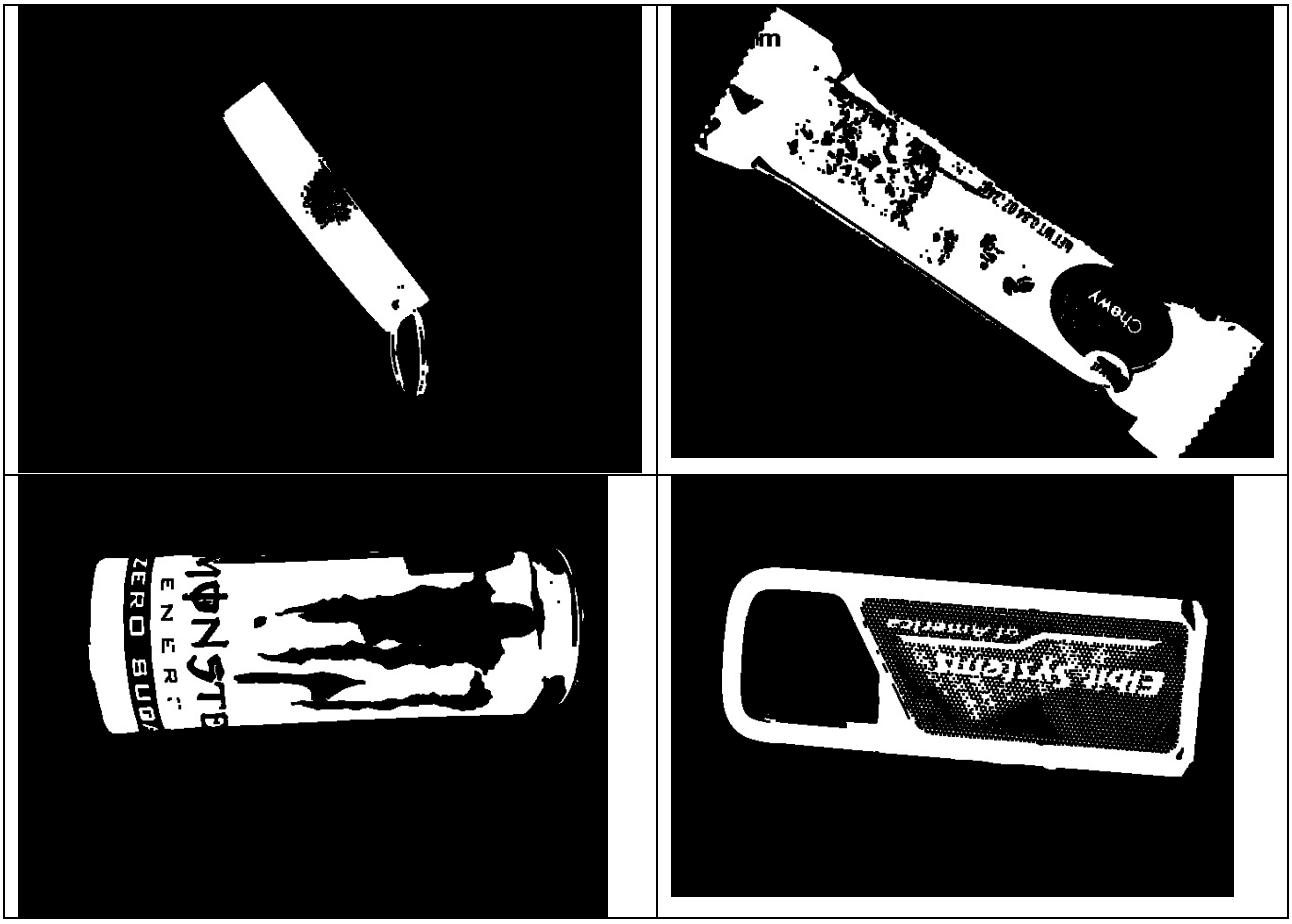
I worked on a project that aims to recognize 2D objects like cutters, action figures, and bracelets that are placed on a white surface. The system is designed to identify these objects irrespective of their position, size, or orientation by using computer vision techniques. It makes use of classification algorithms like K-Nearest Neighbours to ensure accurate identification of the objects, and it can work with both still images and real-time video. Unfortunately, I had to take two days off for personal reasons related to travel.

THRESHOLD FUNCTION:

In my project, I implemented a technique called one-sided thresholding to distinguish the foreground object from the white background. Before that, I applied a bilateral filter to smoothen the image. The purpose of this filter was to eliminate noise while maintaining the image's edges, which the Gaussian filter cannot do. After these steps, I generated binary images, which helped me to obtain the desired output.

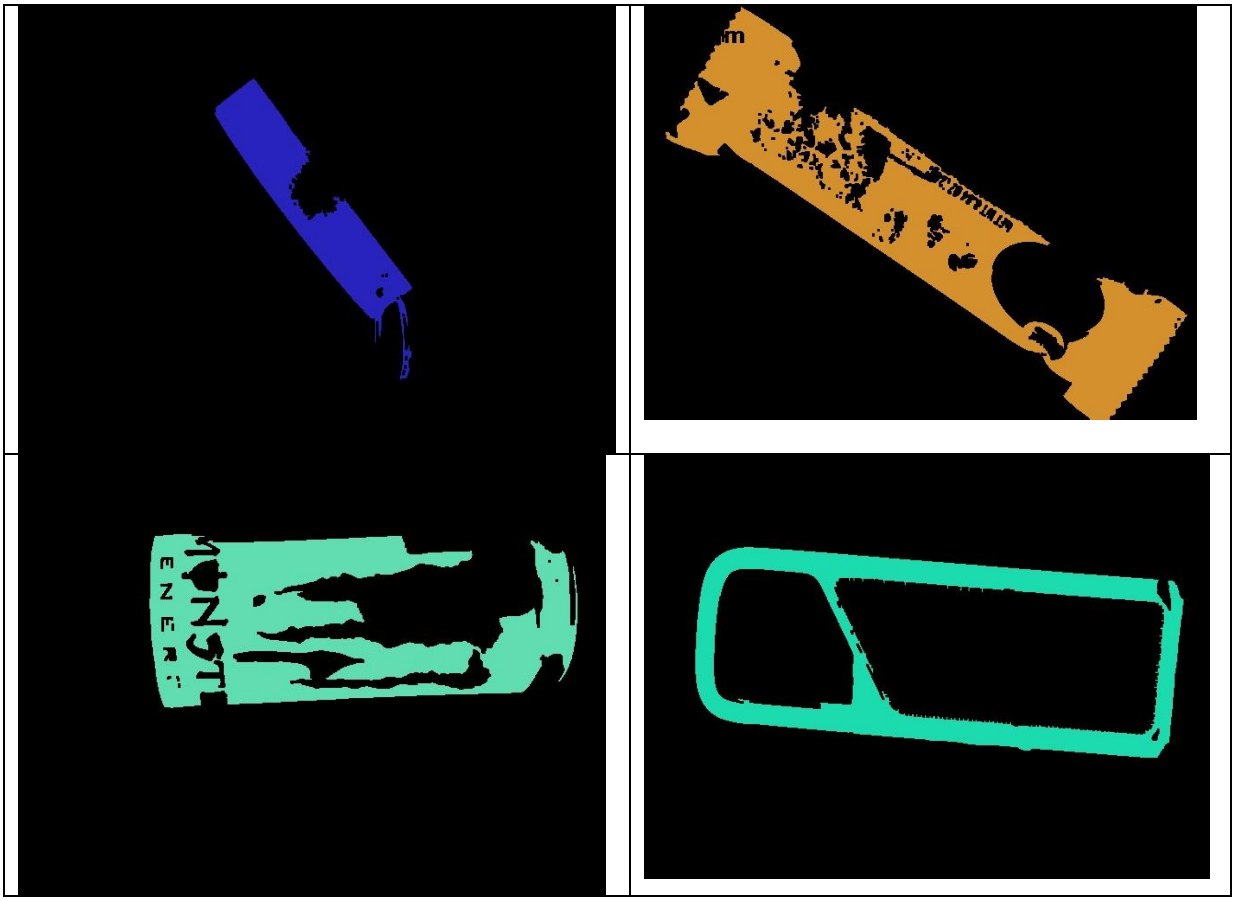
CLEAN IMAGES:

When I applied one-sided thresholding to my images, I noticed that there were holes in the foreground object, which needed to be fixed. To solve this, I used the dilation function from OpenCV, but it resulted in the loss of the object's original shape. In order to bring it back to its original size and shape, I applied the erosion morphological function from OpenCV. This helped me to achieve the desired result, which is shown below.



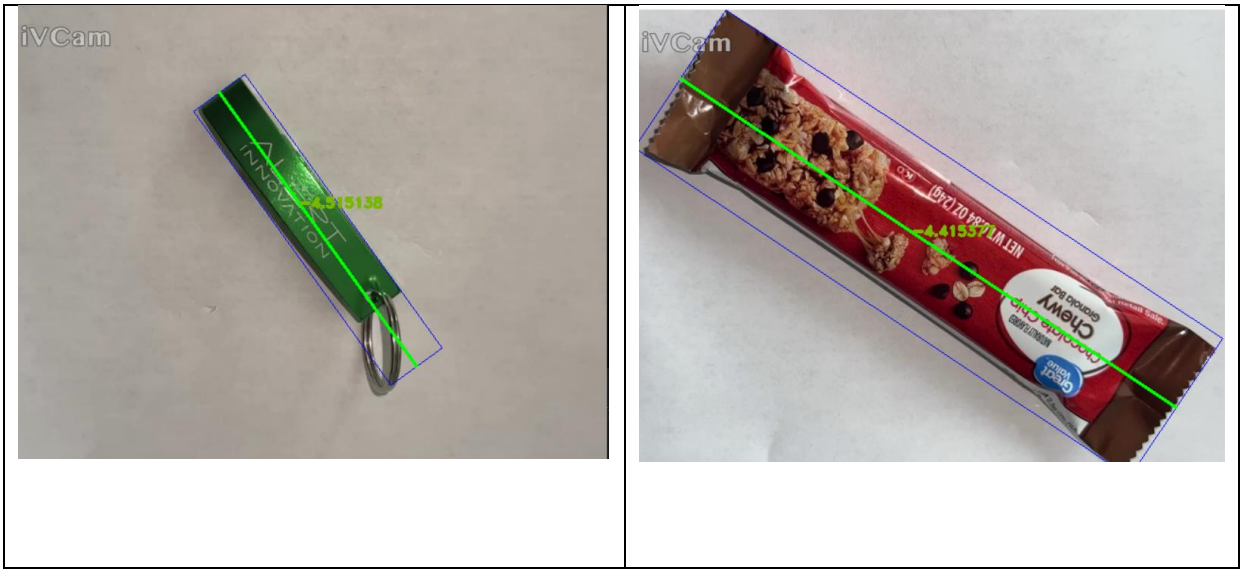
SEGMENT FUNCTION:

To identify each region of the image, I performed a connected component analysis. However, I excluded regions smaller than 1000 pixels. Once I had found all the regions, I displayed each one in a unique color so that they could be easily distinguished.



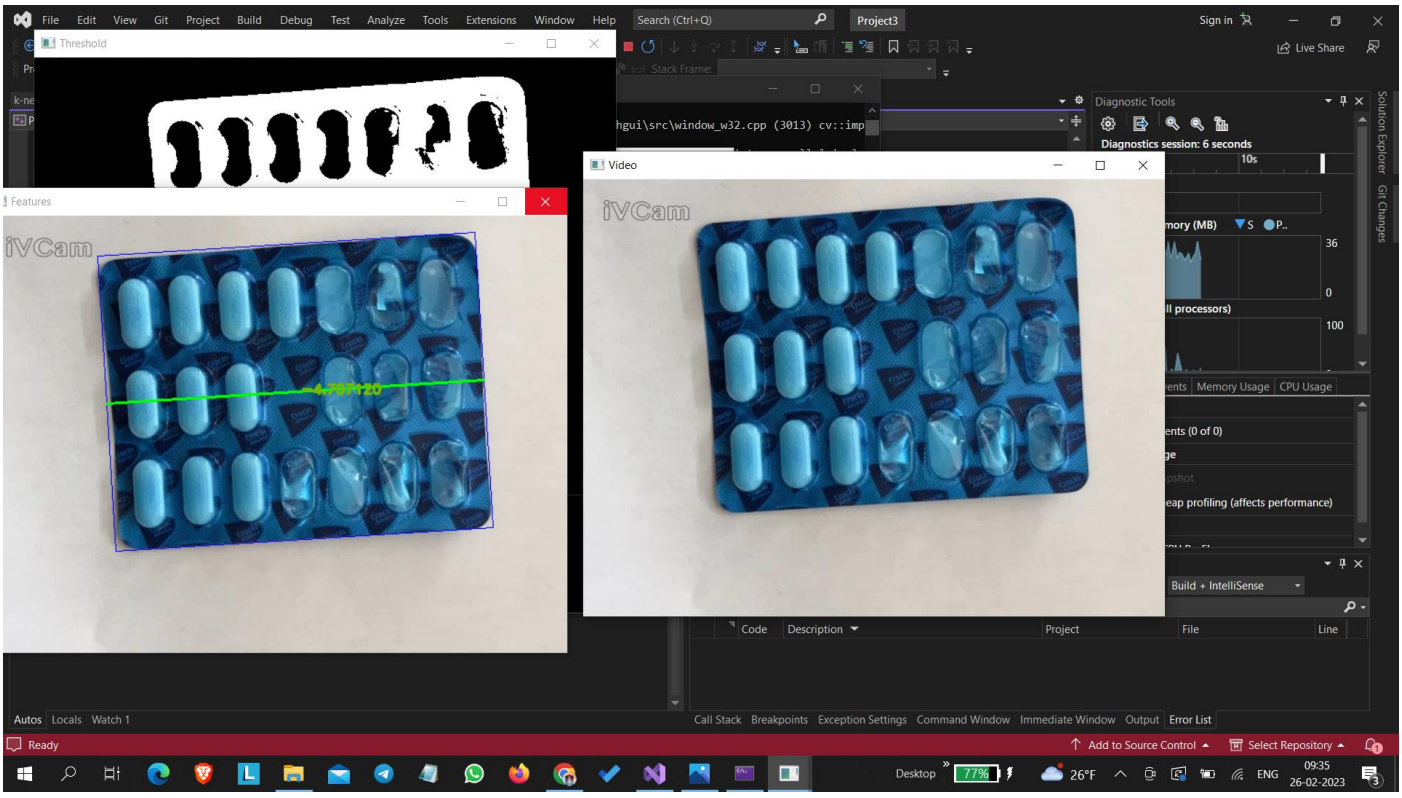
FEATURE EXTRACTION:

In the next stage of my project, I computed a set of moments using OpenCV on the regions I obtained earlier. These moments enabled me to determine the the axis of least central moment and the oriented bounding box. Additionally, I calculated the hu moments, which are invariant to translation, rotation, and scale. I utilized these hu moments as features, and I displayed one of them on the output frame.



TRAINING FUNCTION:

During the video playback, the user has the option to press the 'T' key to capture a snapshot of the current frame. Once captured, the program calculates the features of the image and prompts the user to label it. The labeled image and its corresponding features are then saved in a CSV file, which can be retrieved later for further use.



CLASSIFICATION:

I captured each frame of my real-time video and gave that as input to my classifier. I used two classifiers for this task. My baseline classifier was a simple calculation of the L2 norm as the distance metric and retrieving the label with the least distance from the image. I also implemented a classifier using K-Nearest Neighbours that returns a label which is the mode of the K nearest neighbors.

ACCURACY EVALUATION:

Accuracy: 61.1%

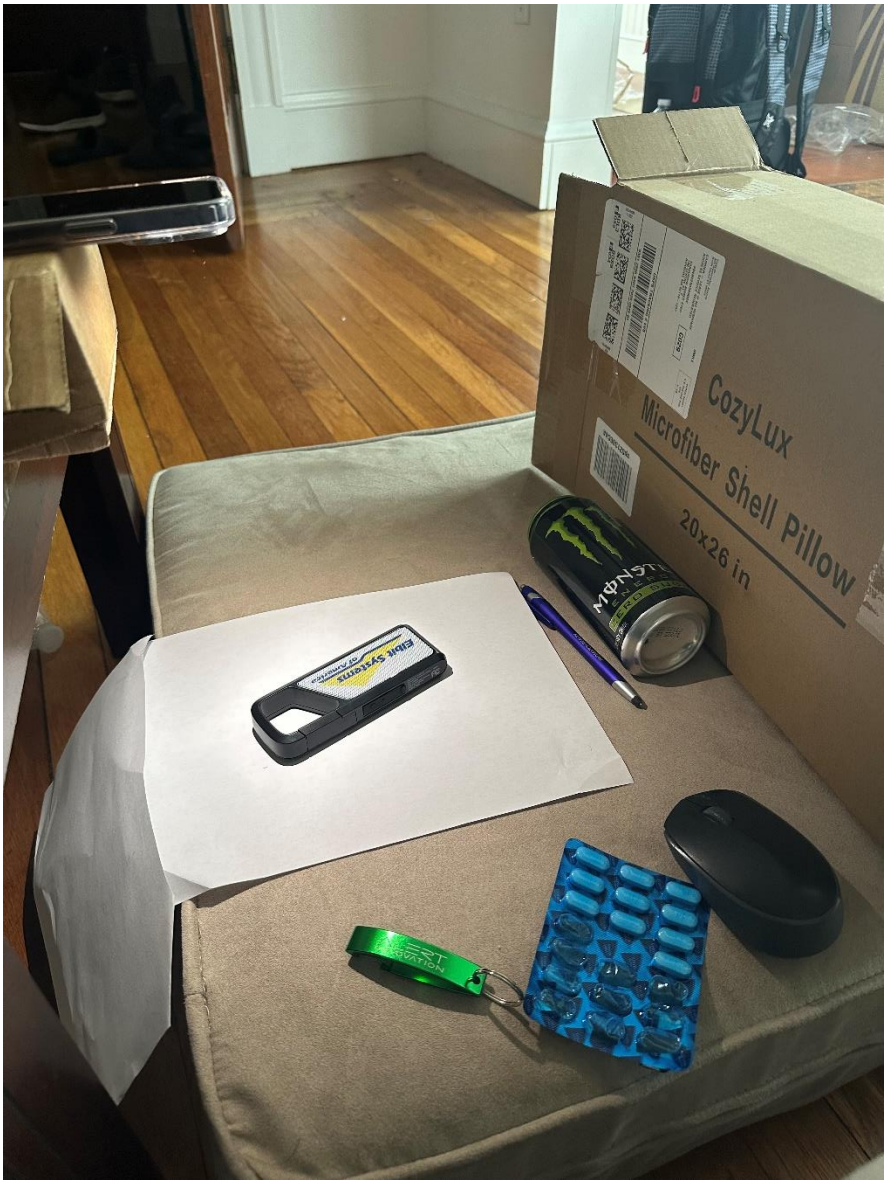
A1

	A	B	C	D	E	F	G	H	I	J	K
1											
2	*	alert_key	candy	key	medicine	mice	pen	soda	speaker	tws	
3	alert_key	1	1	0	0	0	0	0	0	0	
4	candy	0	1	0	0	0	0	0	0	0	
5	key	0	0	0	0	0	0	0	0	0	
6	medicine	0	0	0	1	0	0	0	0	0	
7	mice	1	0	0	0	2	0	0	0	0	
8	pen	0	0	1	0	0	1	0	0	0	
9	soda	0	0	1	1	0	1	2	0	0	
10	speaker	0	0	0	0	0	0	0	1	0	
11	tws	0	0	0	0	0	0	0	1	2	
12											
13											
14											
15											

LIVE VIDEO LINK:

Detection.avi

THINGS USED and SETUP:



I managed to use my phone camera as an external camera by connecting it via a network app, but it was lagging. To enhance the image quality, I used the phone's level-5 light and placed a box in front of the camera to avoid glare and reflections. It's satisfying to find a workaround solution to make things work despite some limitations.

EXTENSIONS:

1) Making menu gui
   I attempted to create a GUI that would allow users to select options from a menu, rather than using the terminal to input commands such as 'T' for training or 'S' for saving. However, I was unable to complete it due to time constraints. I encountered an issue where I needed to take user input in the menu, but I couldn't find a suitable solution for passing this input. I am hoping to complete this task in the future.

2) Unknown object:
   During the development of my project, I was excited to add a new feature that would allow the program to recognize whether an object belonged to the database. Although it was challenging, I successfully implemented a K Nearest Neighbour classifier to determine the distance between the object and the database. If the distance exceeded a certain threshold, I programmed the system to classify the object as an unknown object.

3) Multiple objects:
   As one more trial extension to my project, I added the capability to recognize multiple objects simultaneously using connected component analysis. I allowed the user to input N, the maximum number of regions to be found. The program then assigned a different color to each of the N regions for easy identification.

What I learned- Experience

This project showed me that classical computer vision techniques like the K Nearest Neighbour algorithm and moments are also effective. I'm learning a lot about computer vision/pattern recognition and C++ with each project.

References/Sources:

Stackoverflow for error

GeeksforGeeks

OpenCV Tutorials: https://docs.opencv.org/master/d9/df8/tutorial_root.html

Bilateral Filtering in OpenCV:
https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#ga9d7064d478c95d60003cf839430737ed

Confusion Matrix in Machine Learning: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

C++ Reference for cos(): https://www.cplusplus.com/reference/cmath/cos/

Invariant Moments for Pattern Recognition: http://www.sci.utah.edu/~gerig/CS7960-S2010/handouts/Hu.pdf

How to Write Data to a CSV File in C++: https://thispointer.com/how-to-write-data-in-a-csv-file-in-c/