

PATTERN RECOGNITION & COMPUTER VISION
PROJECT 3: REAL TIME 2D OBJECT RECOGNITION
BASIL REJI & KEVIN SANI

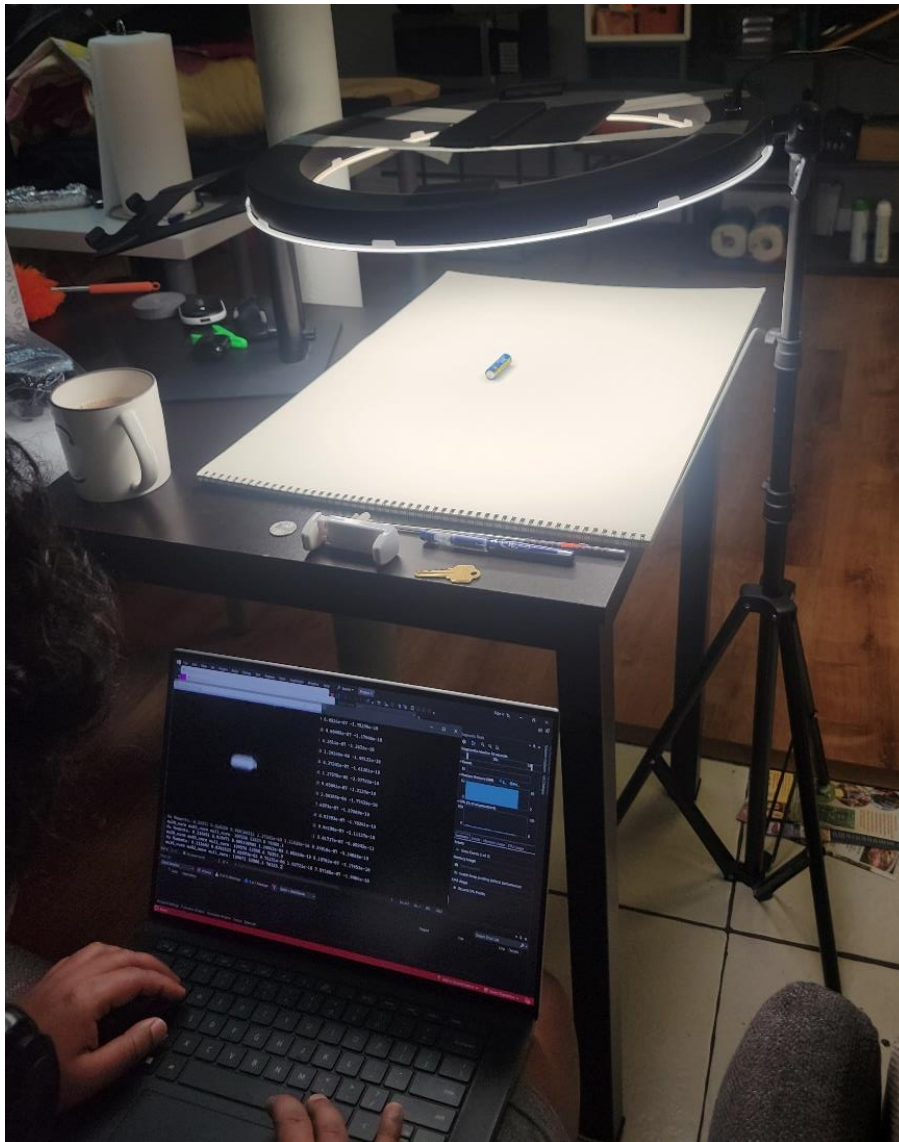
Description:

The project helped us study various aspects of 2D object recognition using computer vision. To distinguish the foreground object of interest from the background, the implemented system uses a phone camera to capture image frames in real-time. After thresholding, it computes the rotation, scale, and translation invariant features of the object, effectively providing us with a cleaned up segmented binary image of the object. The segmentation two-pass approach has an explicitly specified function in the provided implementation, whereas OpenCV library functions are used throughout.

To enter the Training or Classification mode, the user is required to enter inputs. Once in this mode, the system can categorize the objects in real-time or compute features for new item labels and add them to the.csv database file. A recurrent while loop is used to accomplish this until the project file is ended. To determine the best match for an item for classification, a K-NN implementation of distance matching in comparison to the original scaled Euclidean distance is also available. In addition, we manually compute the confusion matrix in order to assess how well the constructed system is working. Paper cutouts of things are included in the database file in addition to actual objects to further test the system.

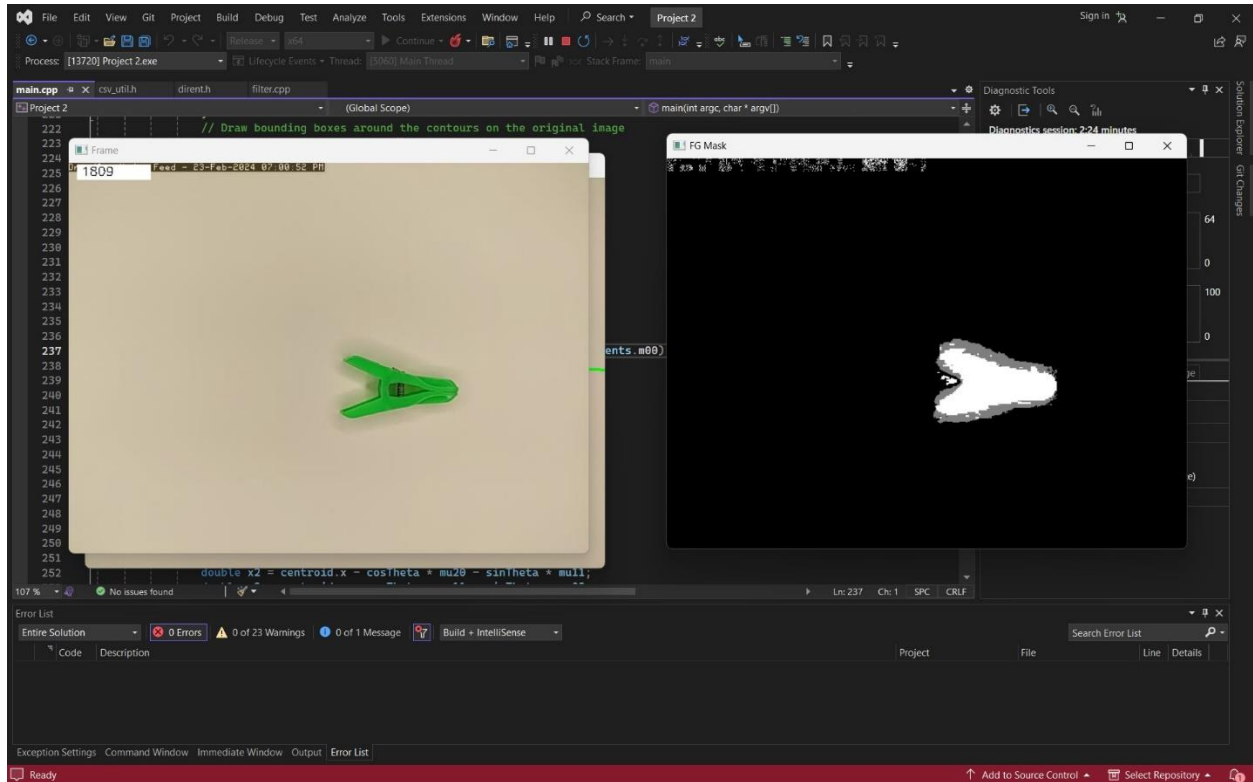
System Setup:



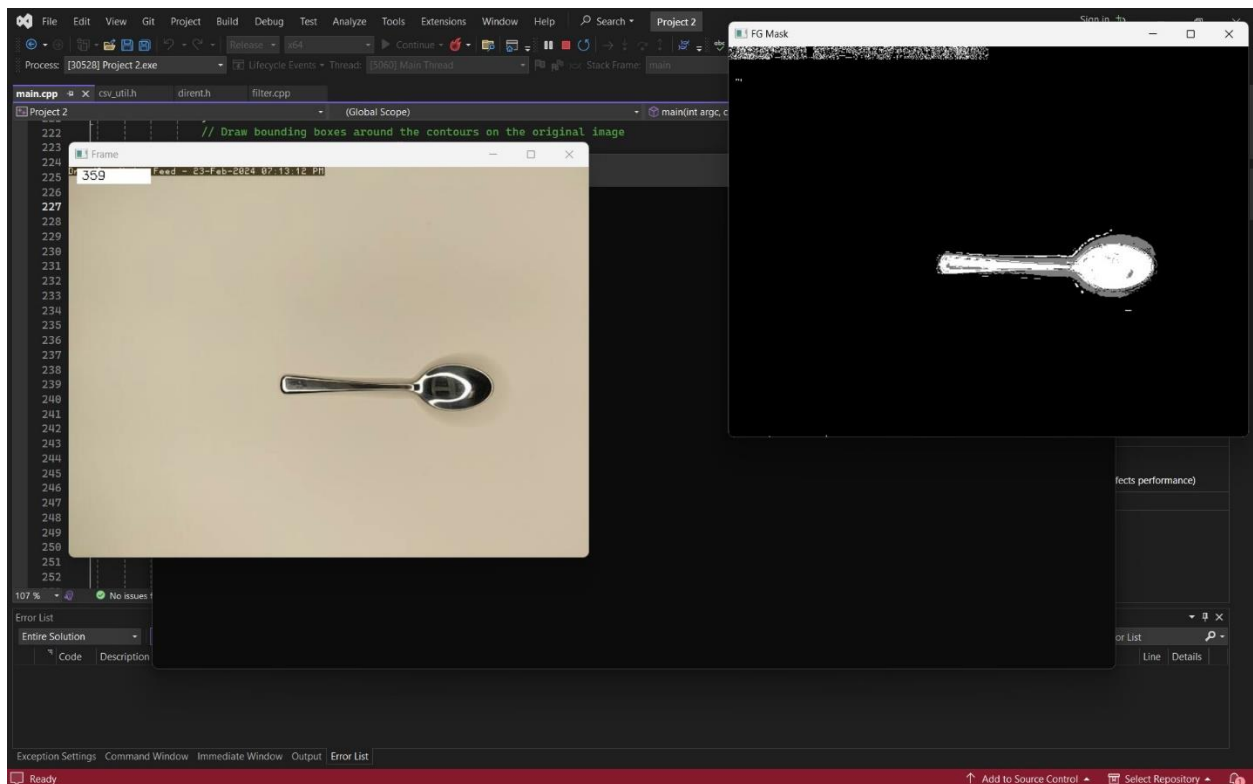


Task 1: Thresholding the Input Video

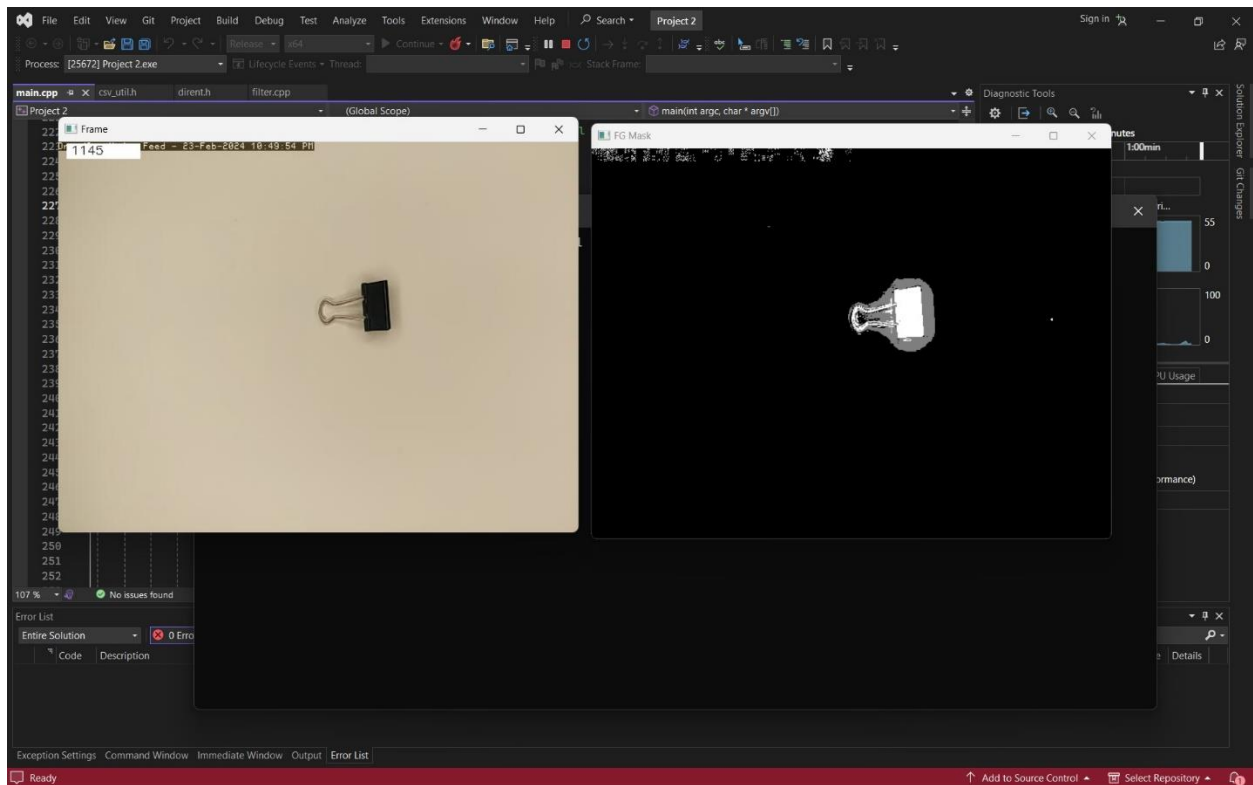
Original input frame and threshold image of a clip



Original input frame and Threshold image of a spoon

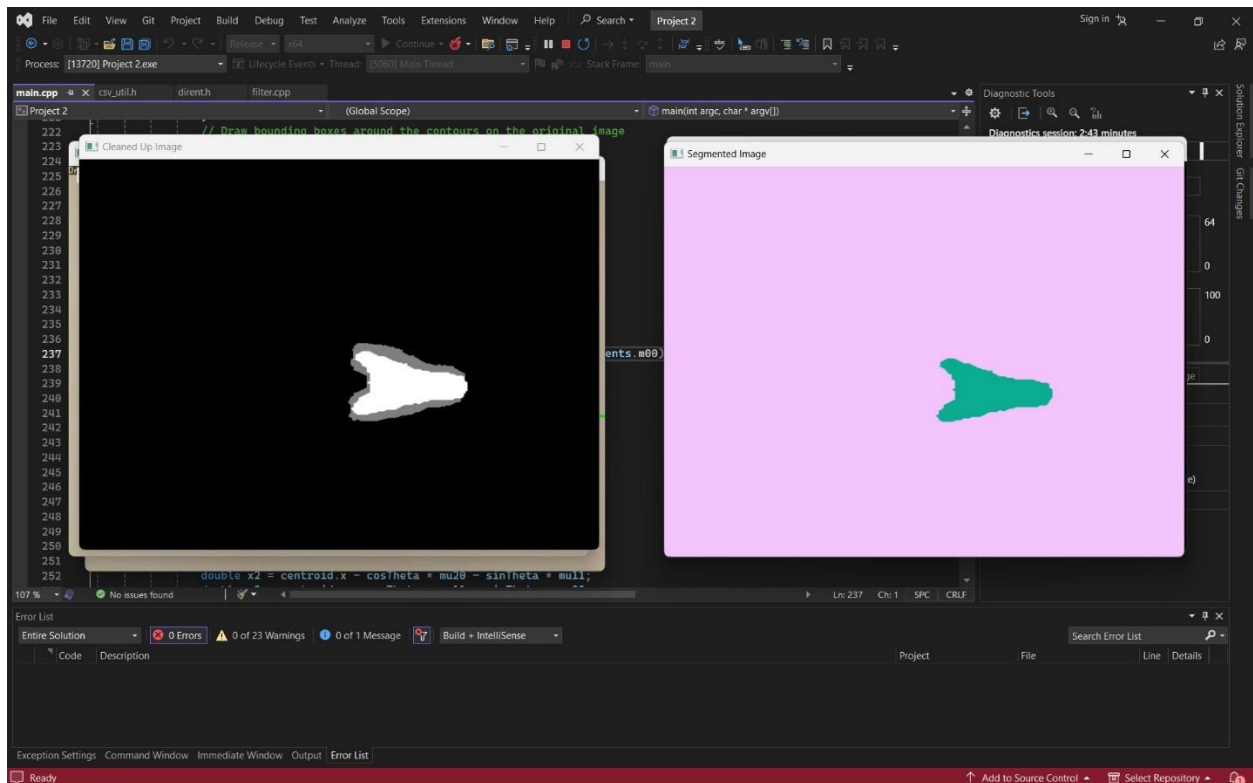


Original input frame and Threshold image of a binding clip

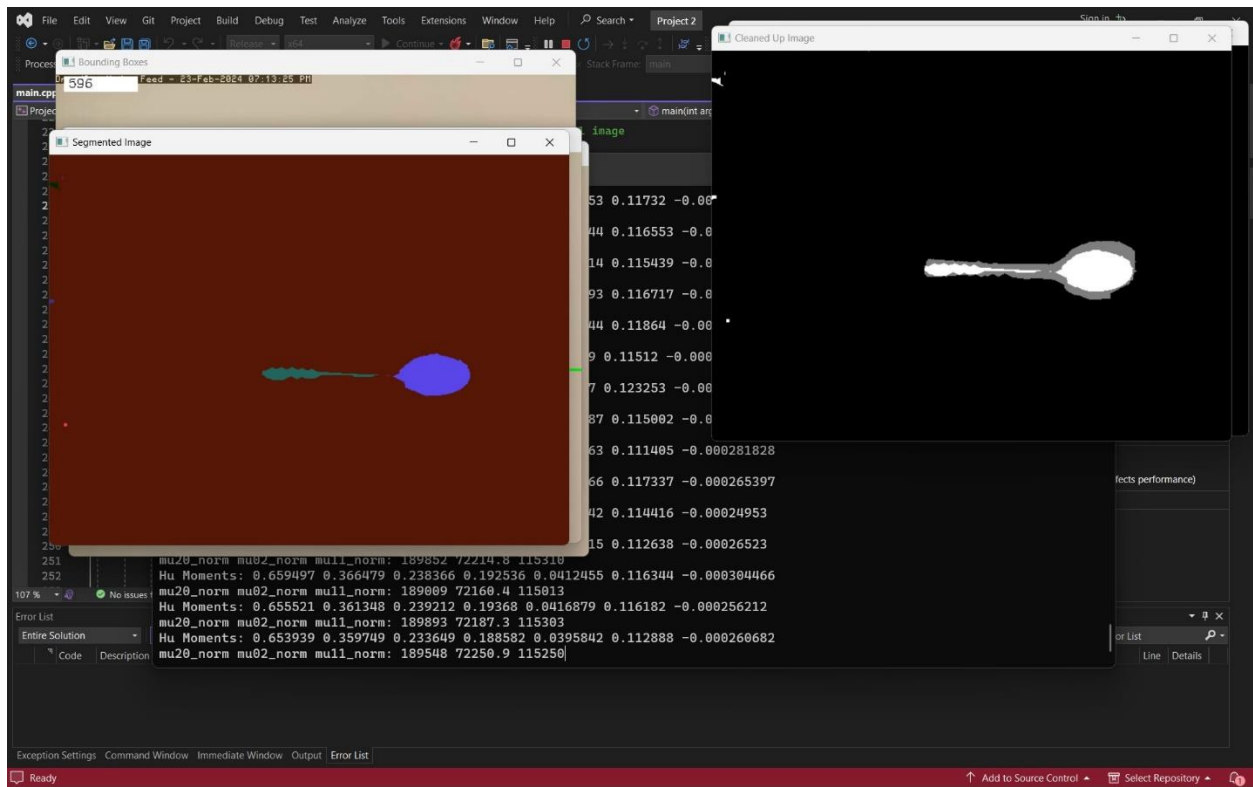


Task 2 and 3: Cleaning Up the Binary Image and Segmenting the Image into Regions

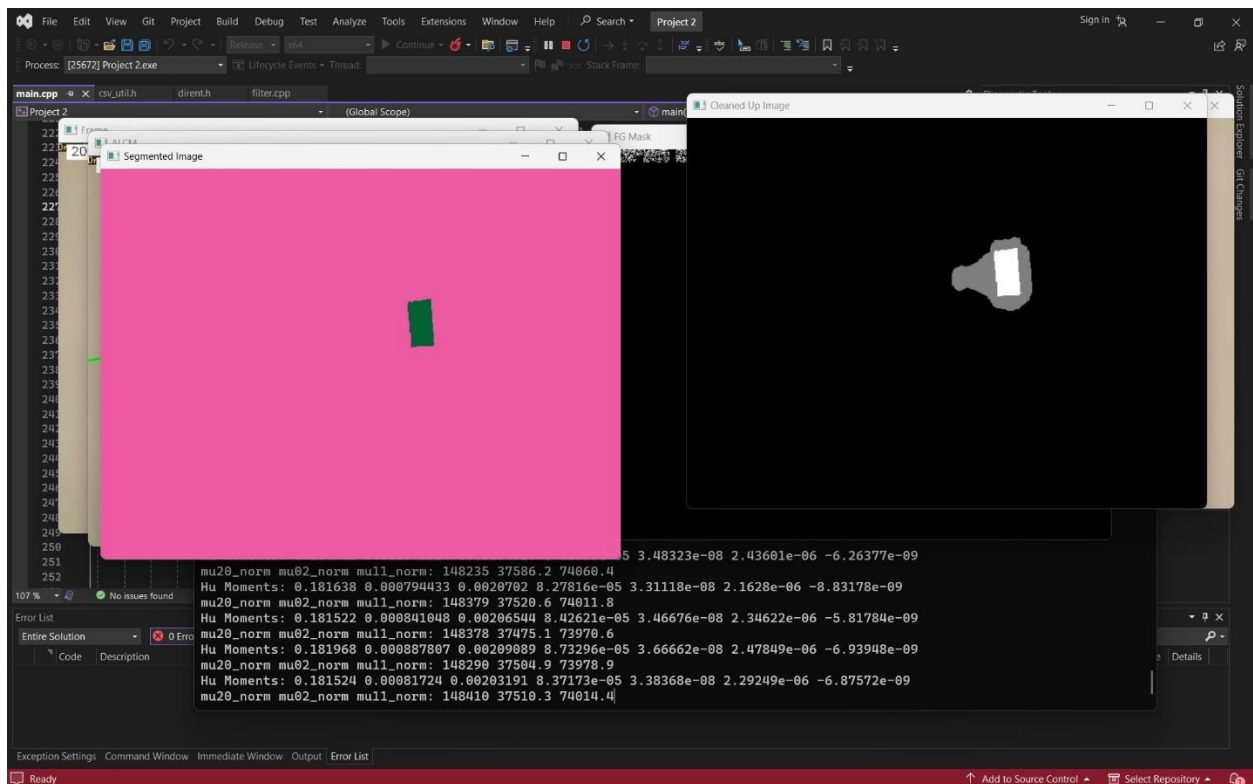
Cleaned up and segmented image of a clip



Cleaned up and segmented image of a spoon

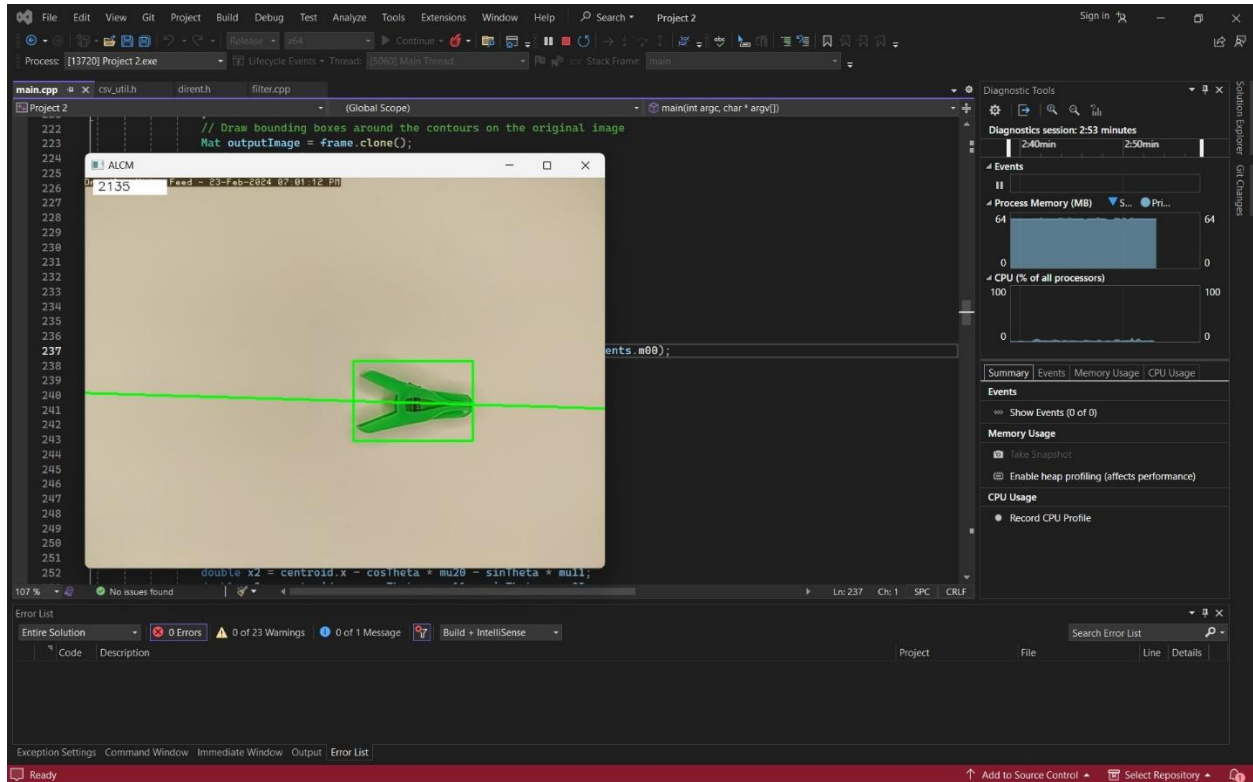


Cleaned up and segmented image of a binding clip

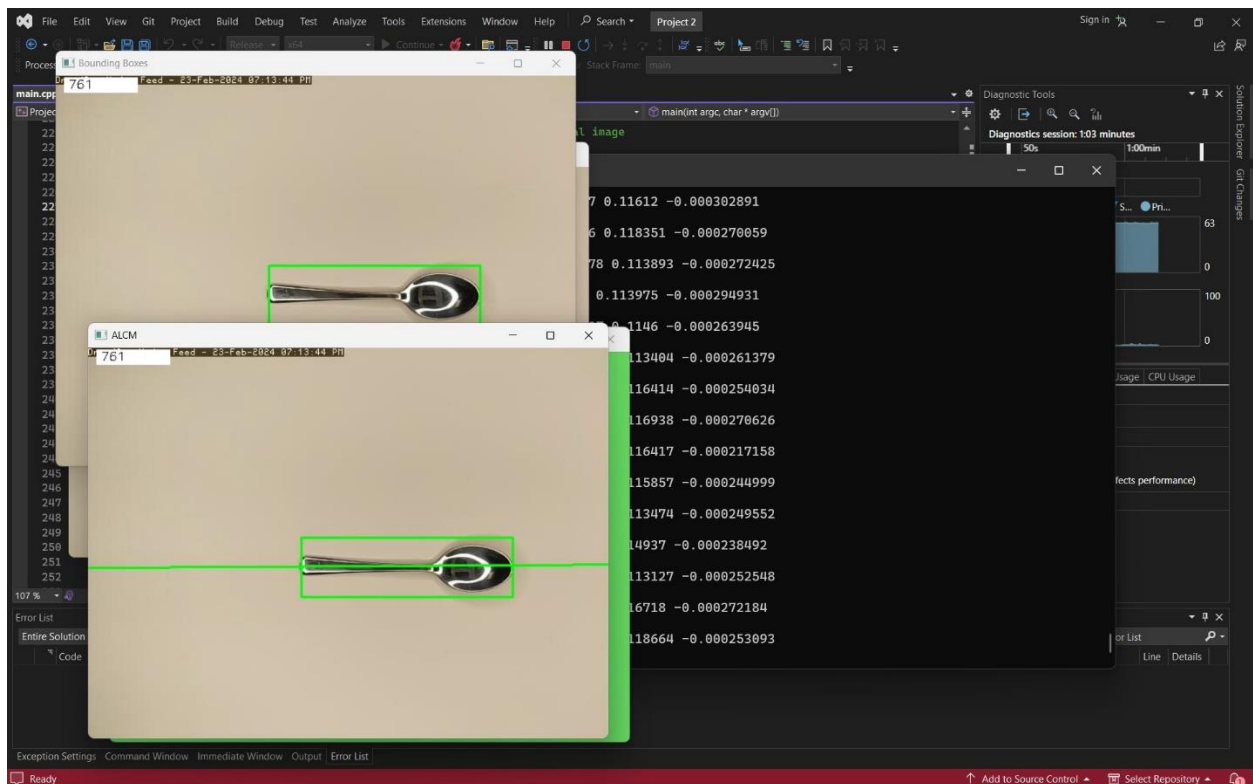


Task 4: Compute Features for Each Major Region

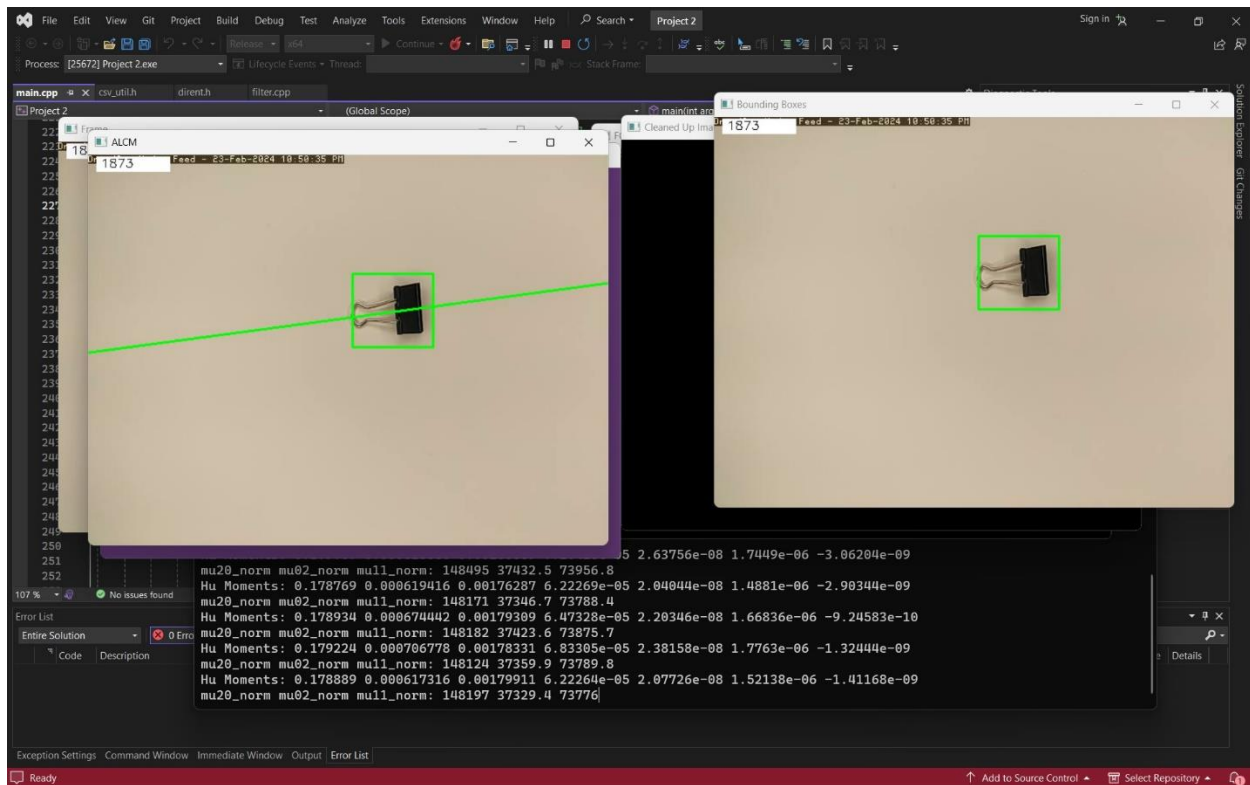
Axis of least central moment and the oriented bounding box of a clip



Axis of least central moment and the oriented bounding box of a spoon



Axis of least central moment and the oriented bounding box of a binding clip



Task 5: Collect Training Data

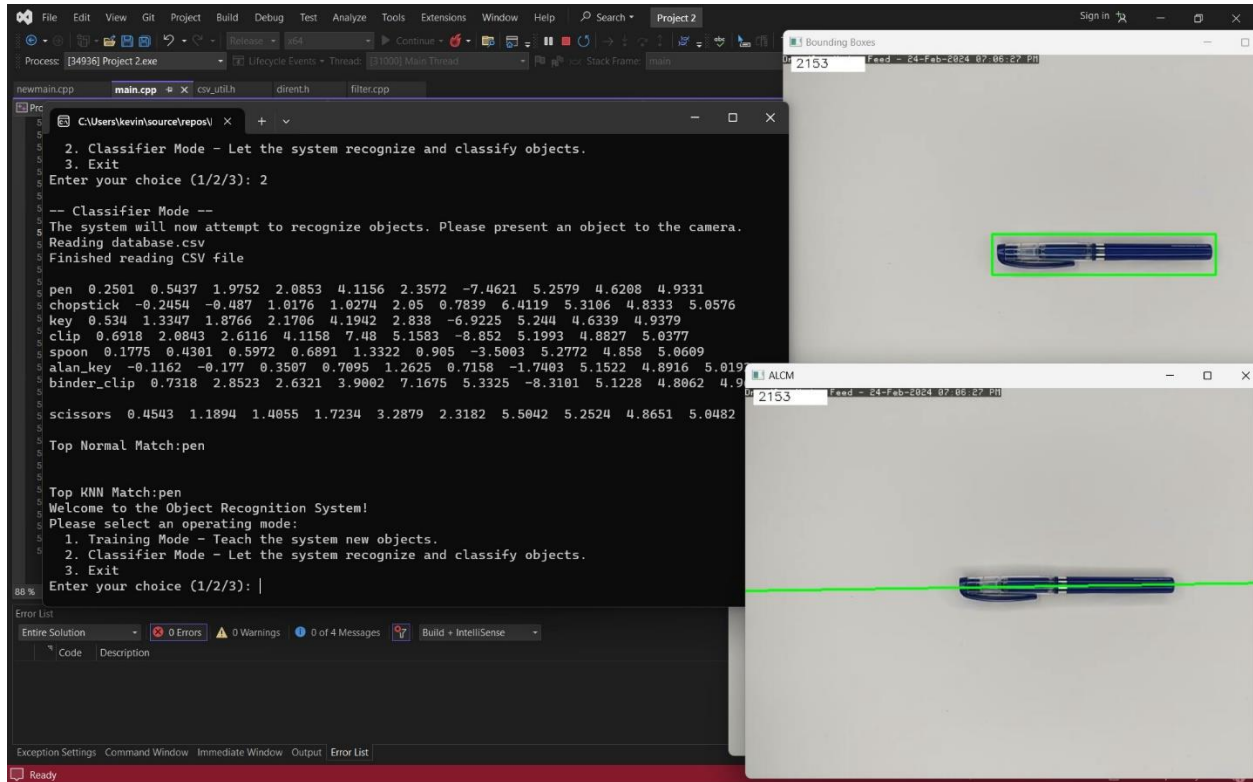
	A	B	C	D	E	F	G	H	I	J	K
1	pen	0.2501	0.5437	1.9752	2.0853	4.1156	2.3572	-7.4621	5.2579	4.6208	4.9331
2	chopstick	-0.2454	-0.487	1.0176	1.0274	2.05	0.7839	6.4119	5.3106	4.8333	5.0576
3	key	0.534	1.3347	1.8766	2.1706	4.1942	2.838	-6.9225	5.244	4.6339	4.9379
4	clip	0.6918	2.0843	2.6116	4.1158	7.48	5.1583	-8.852	5.1993	4.8827	5.0377
5	spoon	0.1775	0.4301	0.5972	0.6891	1.3322	0.905	-3.5003	5.2772	4.858	5.0609
6	alan_key	-0.1162	-0.177	0.3507	0.7095	1.2625	0.7158	-1.7403	5.1522	4.8916	5.0192
7	binder_clip	0.7318	2.8523	2.6321	3.9002	7.1675	5.3325	-8.3101	5.1228	4.8062	4.9622
8	scissors	0.4543	1.1894	1.4055	1.7234	3.2879	2.3182	5.5042	5.2524	4.8651	5.0482

In the training phase of the system, users are greeted with a heading offering a choice between training and classification modes. Upon choosing training mode, the user needs to provide a label for the object they wish to train the system on. Following the entry of the object name and pressing the enter key, the system captures and displays the initial color image alongside the foreground mask image of the object.

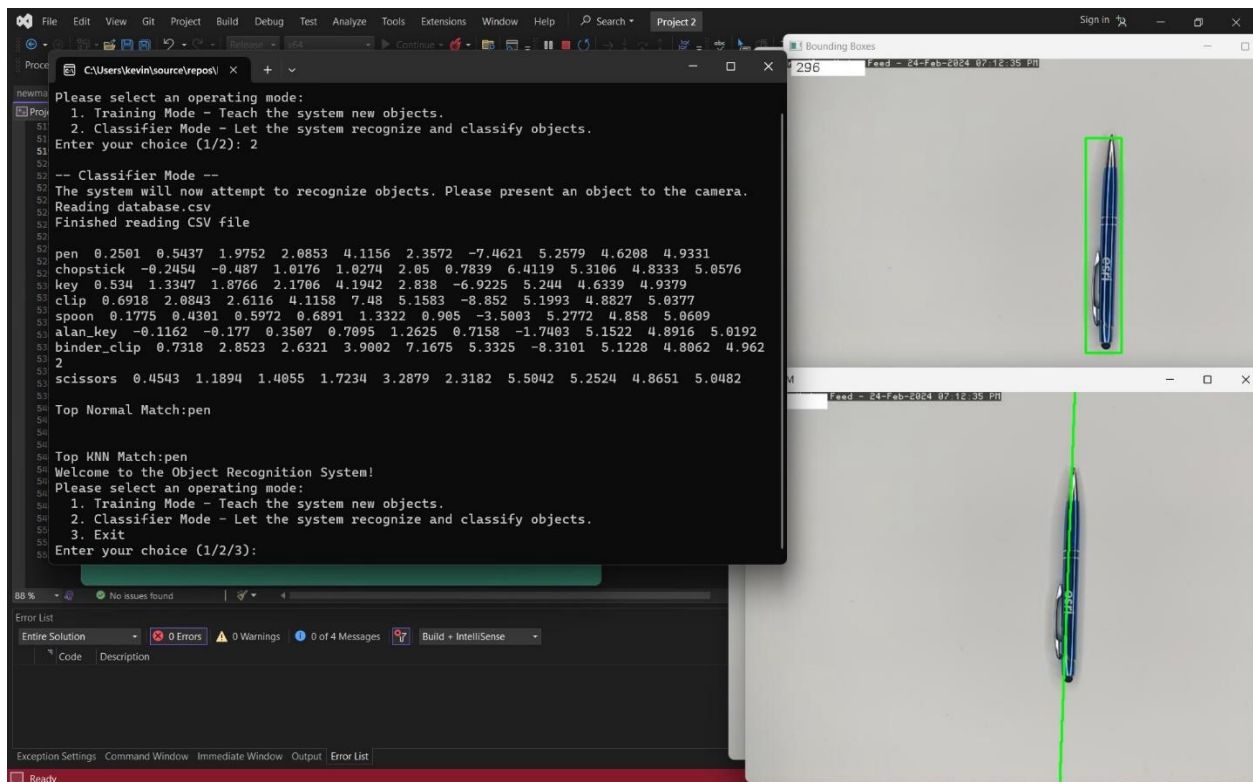
The system identifies the initial frame as the background, which will be subtracted from subsequent frames containing the object. Once the frame stabilizes the user needs to place the object within the camera's field of view and adjust until it is properly focused. Moving on the system performs and shows the result of morphological operations for image cleanup, drawing bounding boxes and axes (Axis of Least Central Moment, ALCM) around the object. The system then calculates and outputs Hu moments and normalized central moments to the terminal. The program appends these calculated features into a CSV file, effectively training the system with the object's data. The system then loops back, allowing users to either add more objects to the training set or switch to classification mode, facilitating a continuous, user-driven training process.

Task 6: Classify New Images

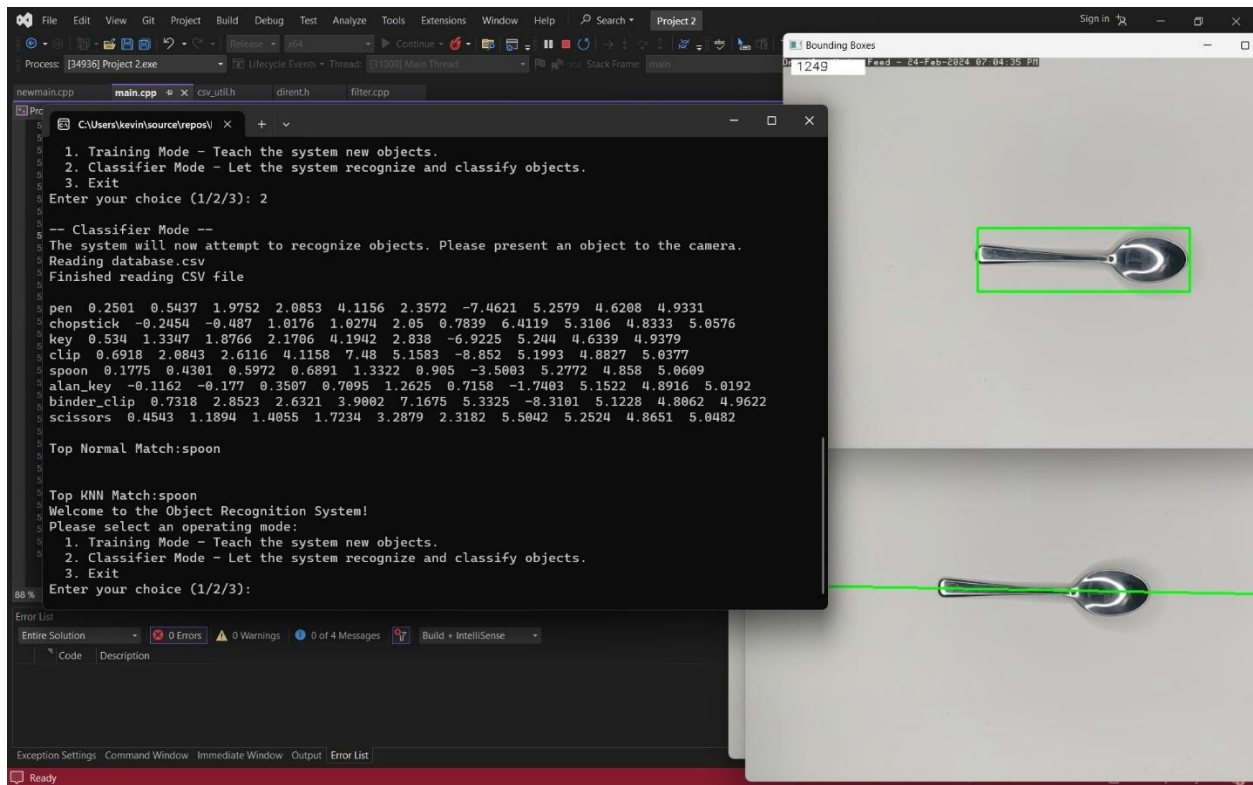
System classifying a pen correctly



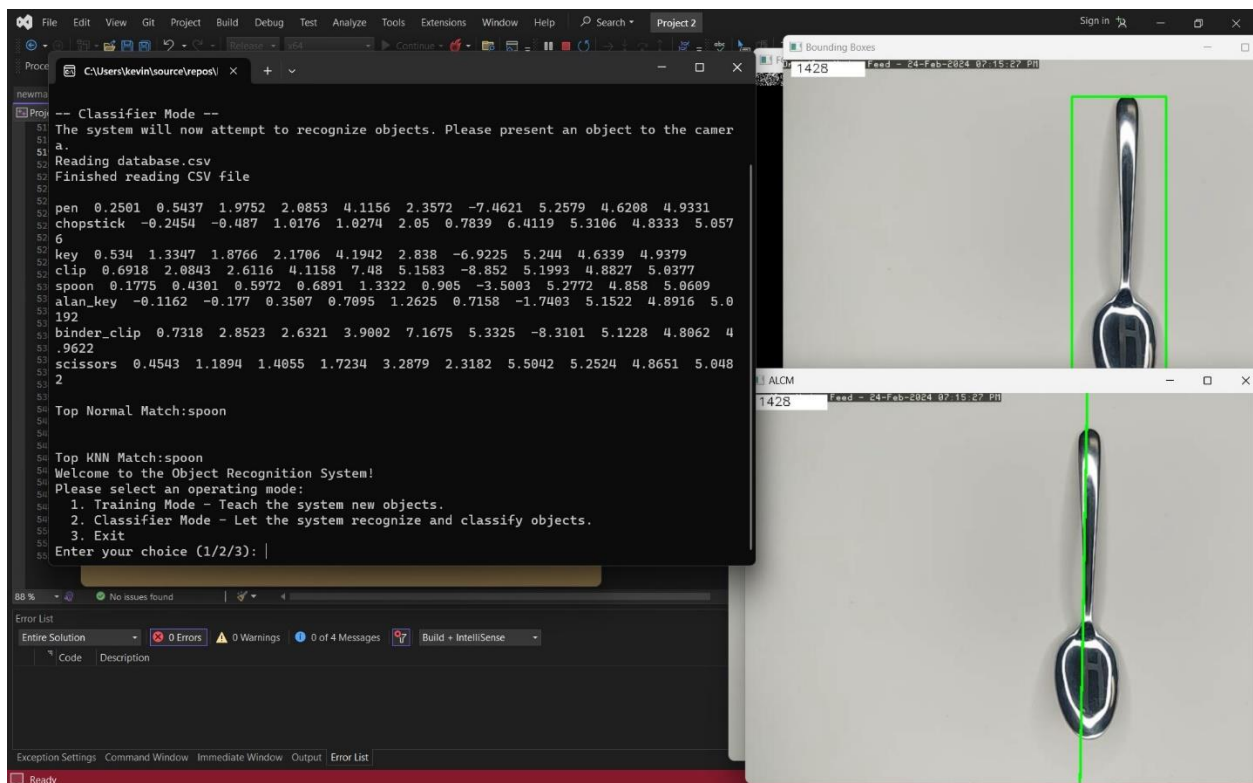
System classifying a different pen in another orientation correctly



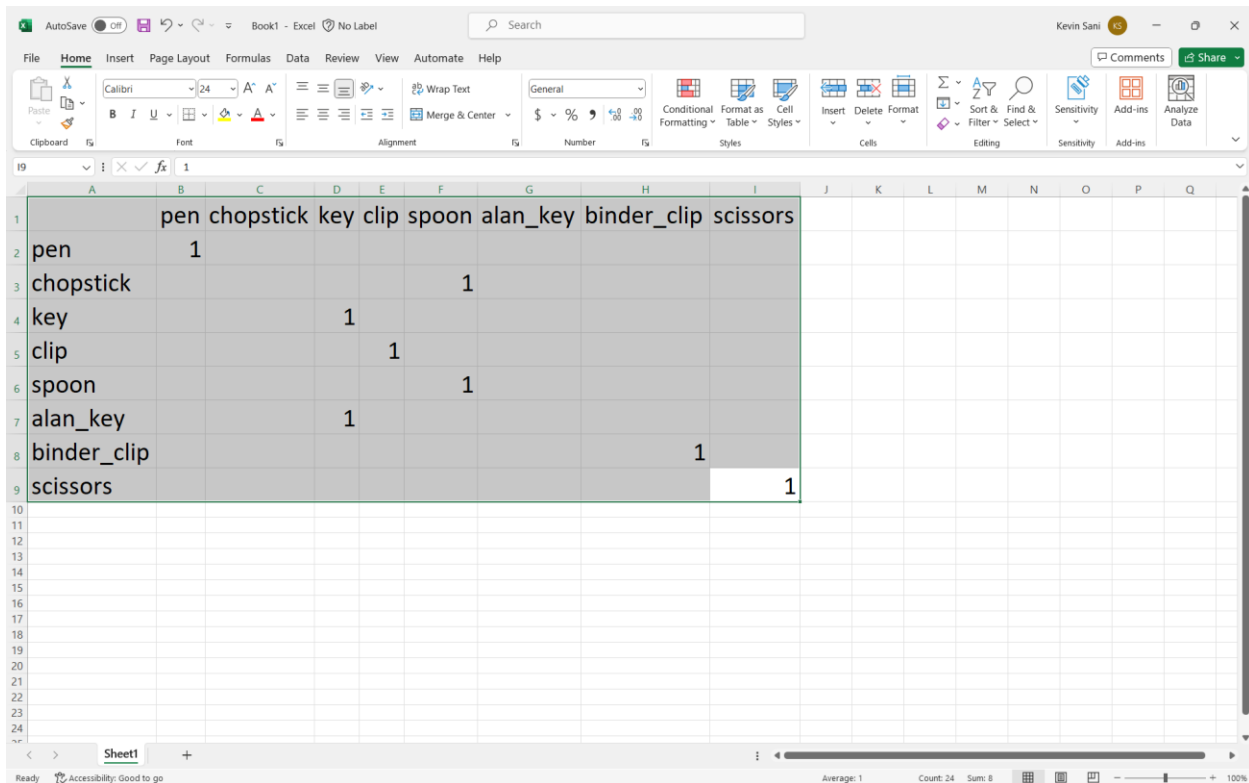
System classifying a spoon correctly



System classifying a different spoon in another orientation correctly



Task 7: Evaluate Performance of the System



	pen	chopstick	key	clip	spoon	alan_key	binder_clip	scissors
pen	1							
chopstick		1						
key			1					
clip				1				
spoon					1			
alan_key						1		
binder_clip							1	
scissors								1

8*8 confusion matrix of different objects. It is seen that only the chopstick and the allen key are misclassified as spoon and key respectively the rest of the objects are classified properly.

Task 8: Capture Demo of Working System

Training Video

<https://drive.google.com/drive/folders/1z3rvuaK2W3vZl6BOFsvNITWIXP3xXRPB?usp=sharing>

Classification Video

https://drive.google.com/drive/folders/1Y-7ptw9LWrxnUj_ts8mZ-X5I-HbuiRi?usp=sharing

Task 9: Implementing a Second Classification Method

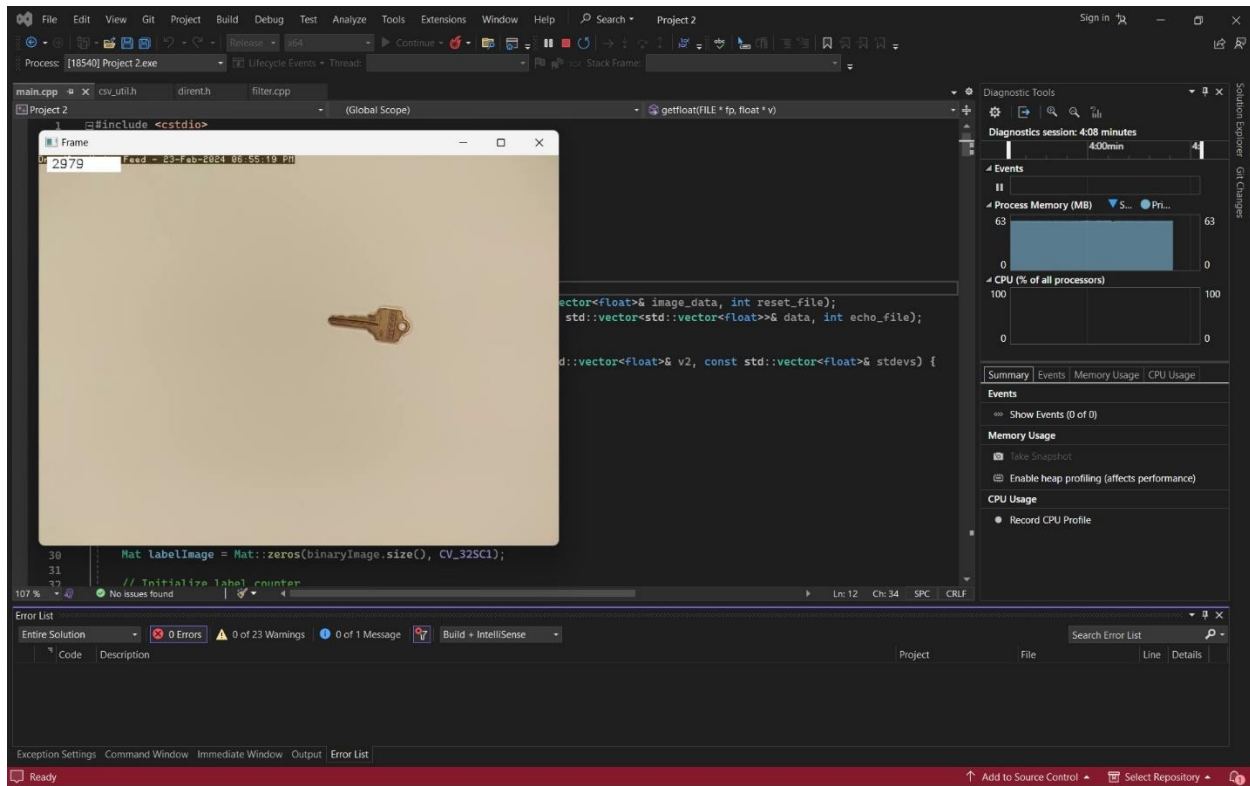
For the second classification method, the choice was to implement K-Nearest Neighbors (KNN) to enhance the object recognition process. This method was selected due to its simplicity and effectiveness in handling multiple training examples per object, without resorting to a voting mechanism for classification. We chose K=2 due to the smaller amount of training data. In contrast to the baseline nearest-neighbor approach, which considers only the closest match, KNN allows for a more robust classification by considering multiple close matches, thereby potentially improving accuracy through a consensus of nearest neighbors.

The implementation involved creating descriptors for both the query and the training data and then using FLANN-based matching to find the 2 nearest neighbors for each query descriptor. The assumption here is that a better match can be determined by looking at the closest two matches and selecting the best one based on the distance metric.

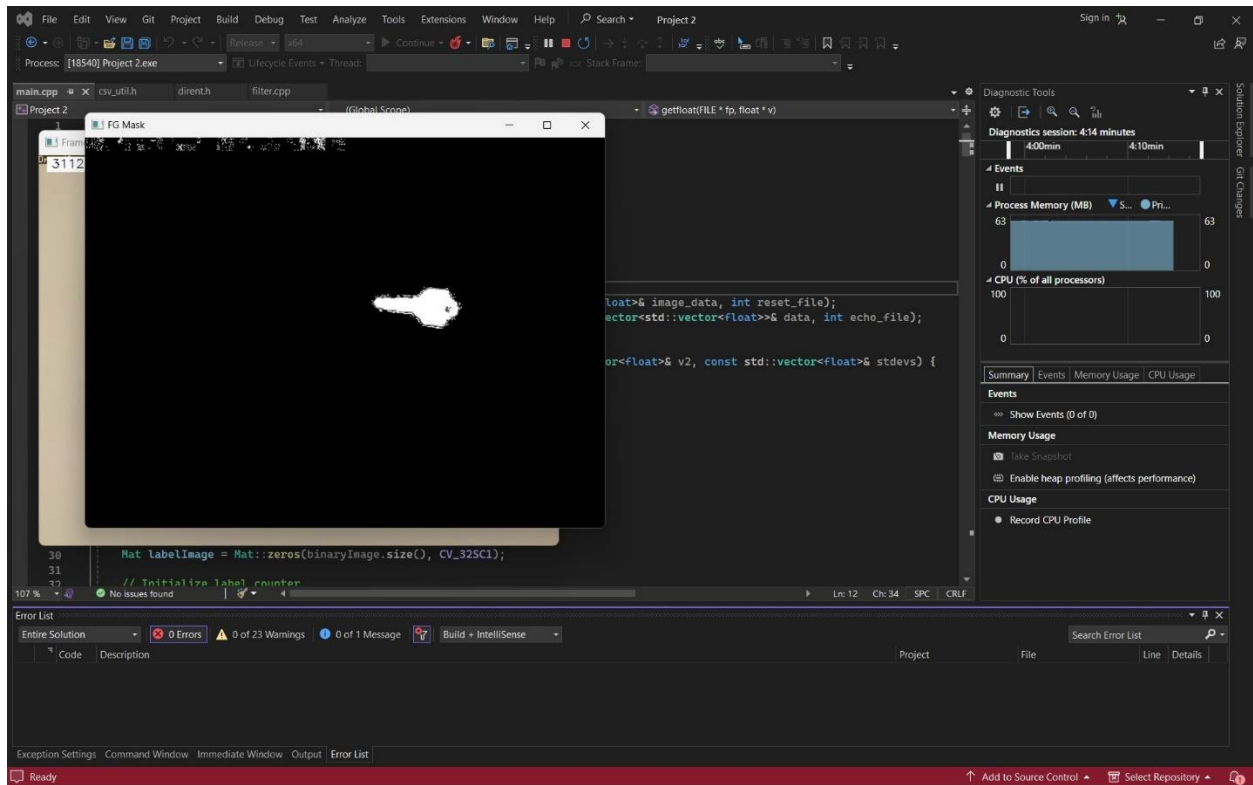
Upon comparing the performance of the baseline system with the KNN classifier, it was observed that both methods returned similar outcomes for the best match. This similarity could be attributed to the nature of the dataset and the distinctiveness of the object features. The KNN approach offers a framework for more nuanced classification decisions, especially in cases where the dataset is larger or the objects have subtle differences, by evaluating multiple nearest neighbors rather than relying on a single closest match.

Extensions

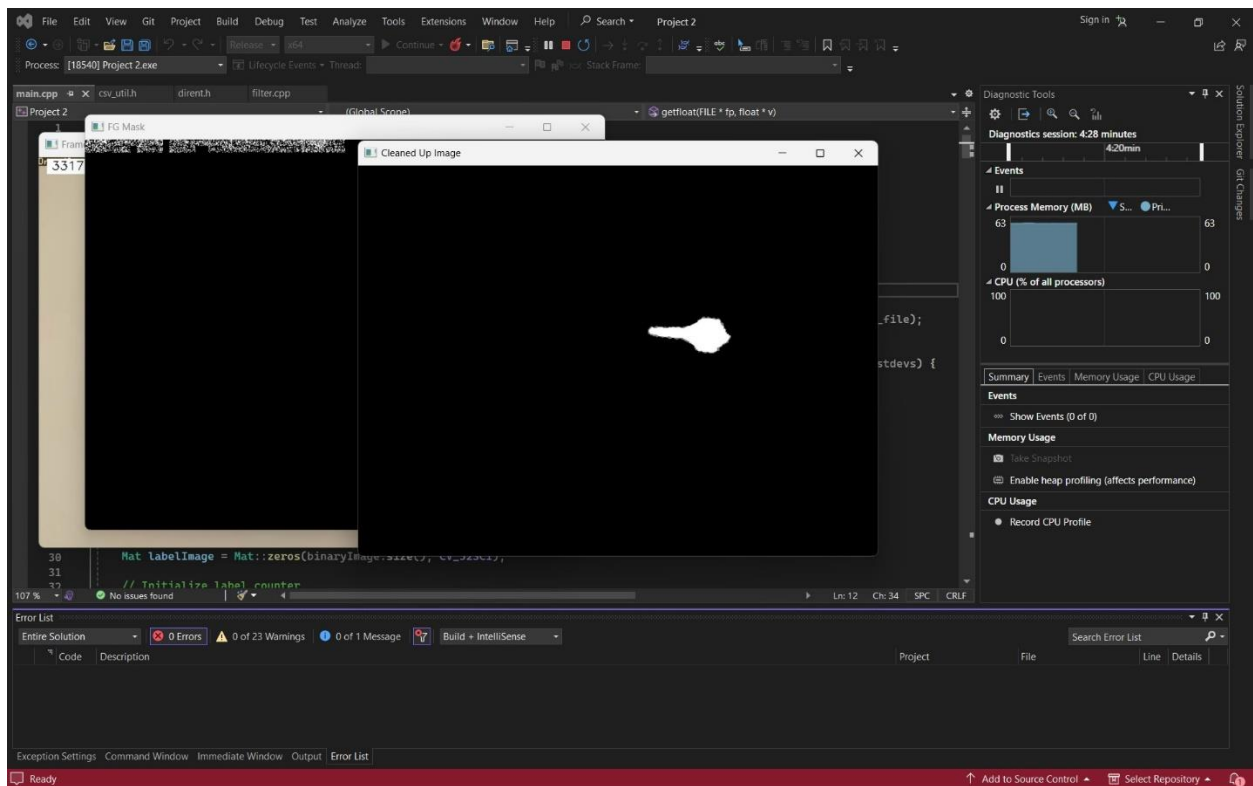
Original input frame image of a key



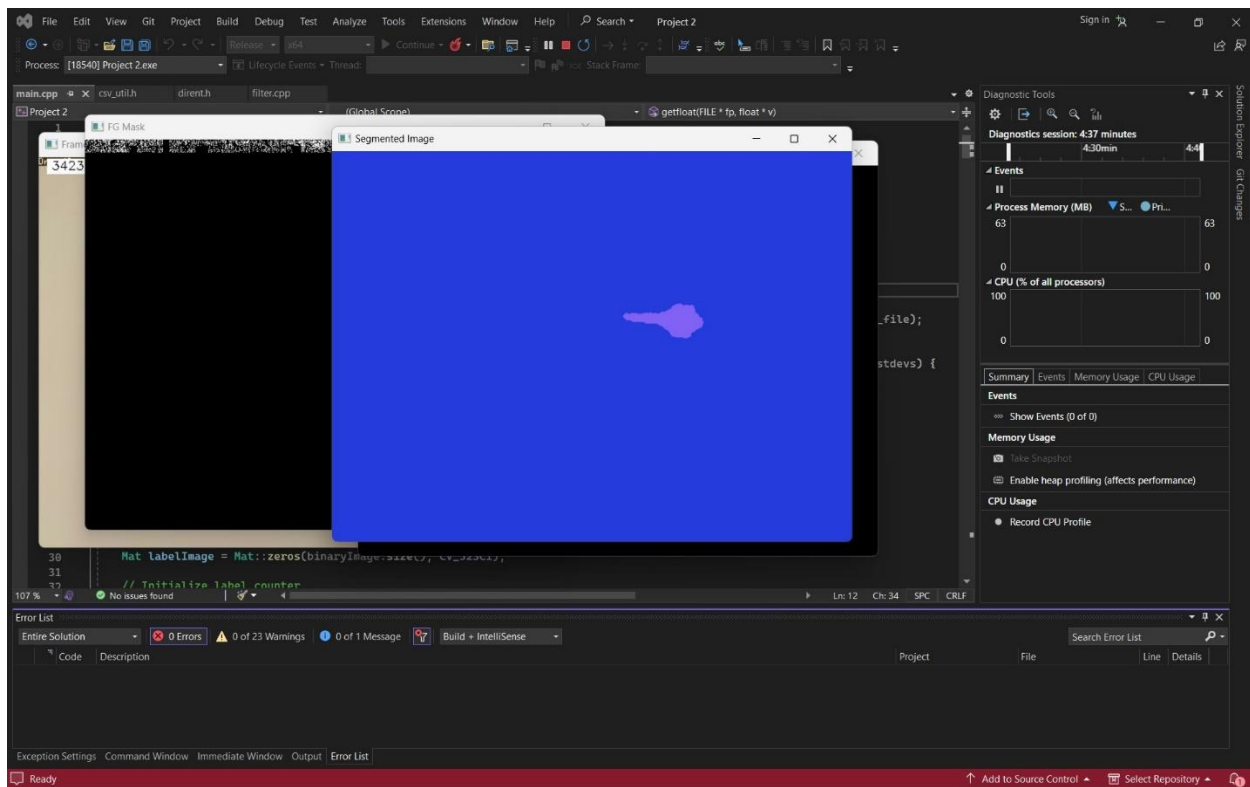
Threshold image of a key



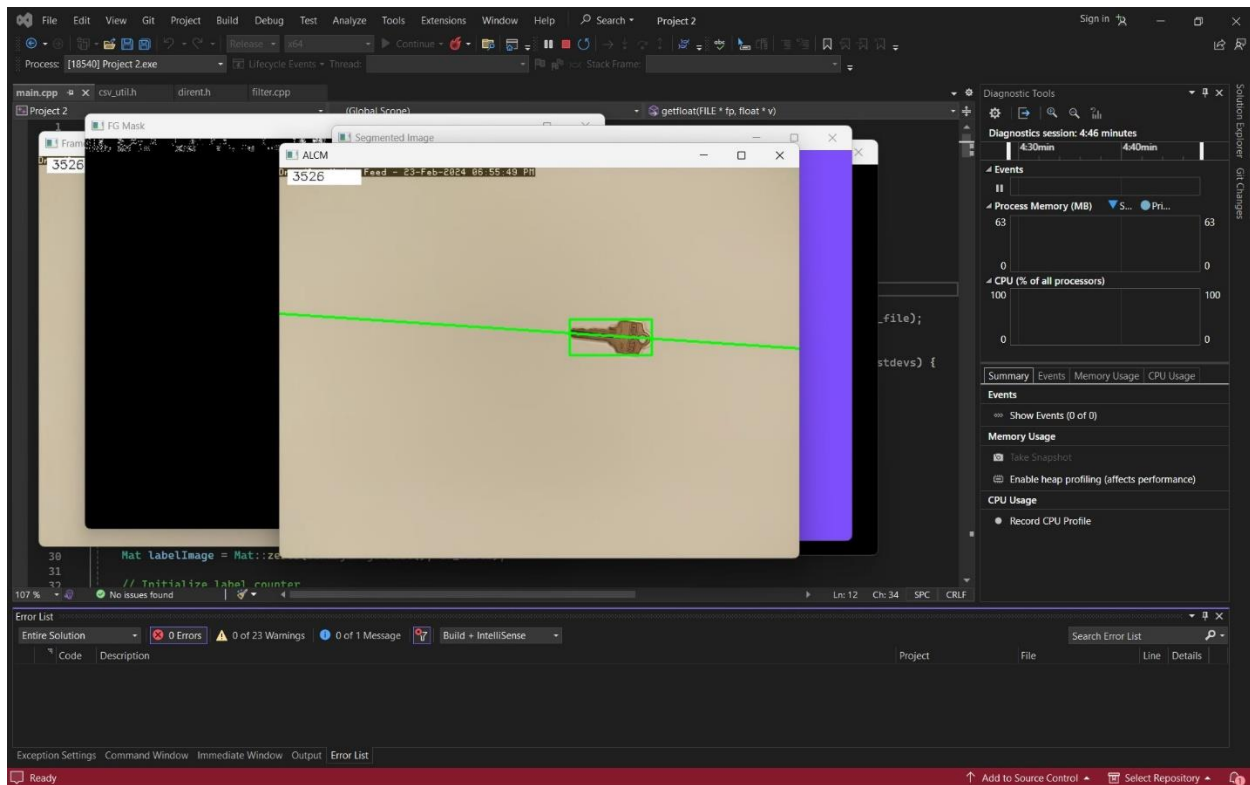
Cleaned Up image of a key



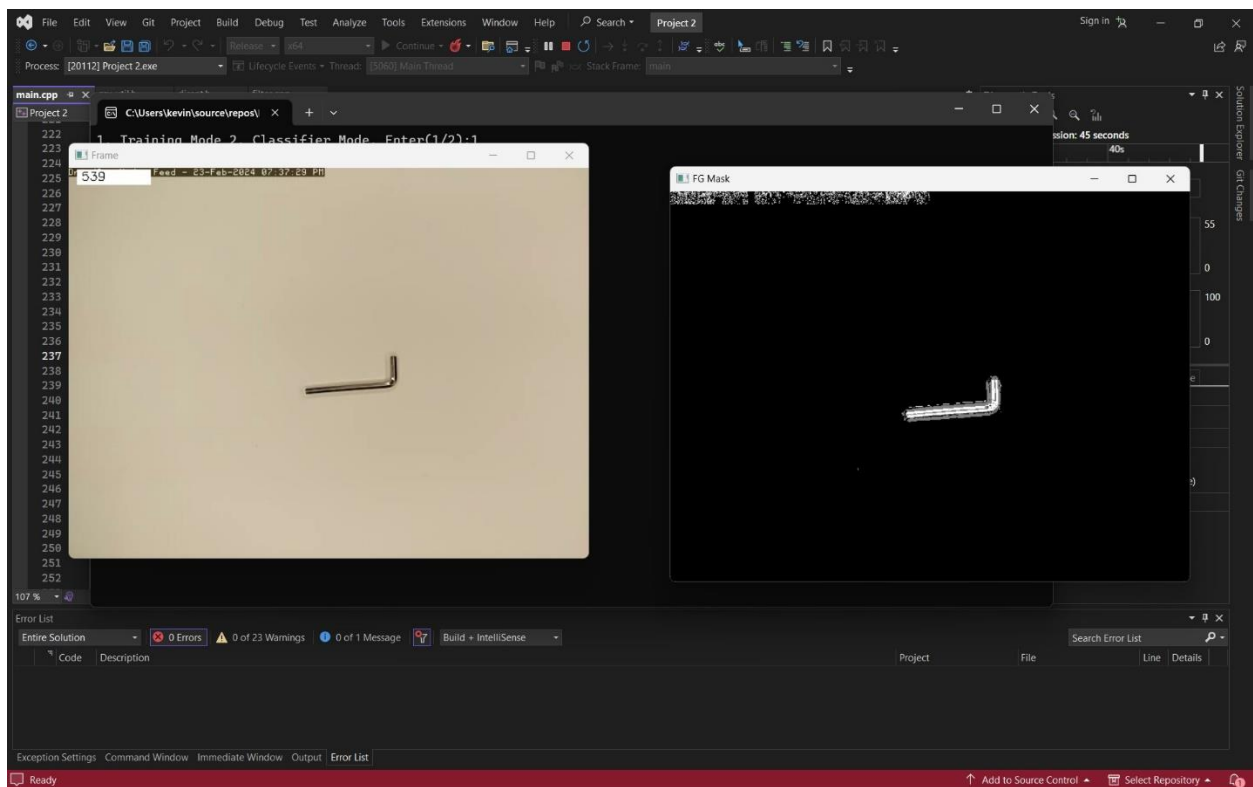
Segmented image of a key



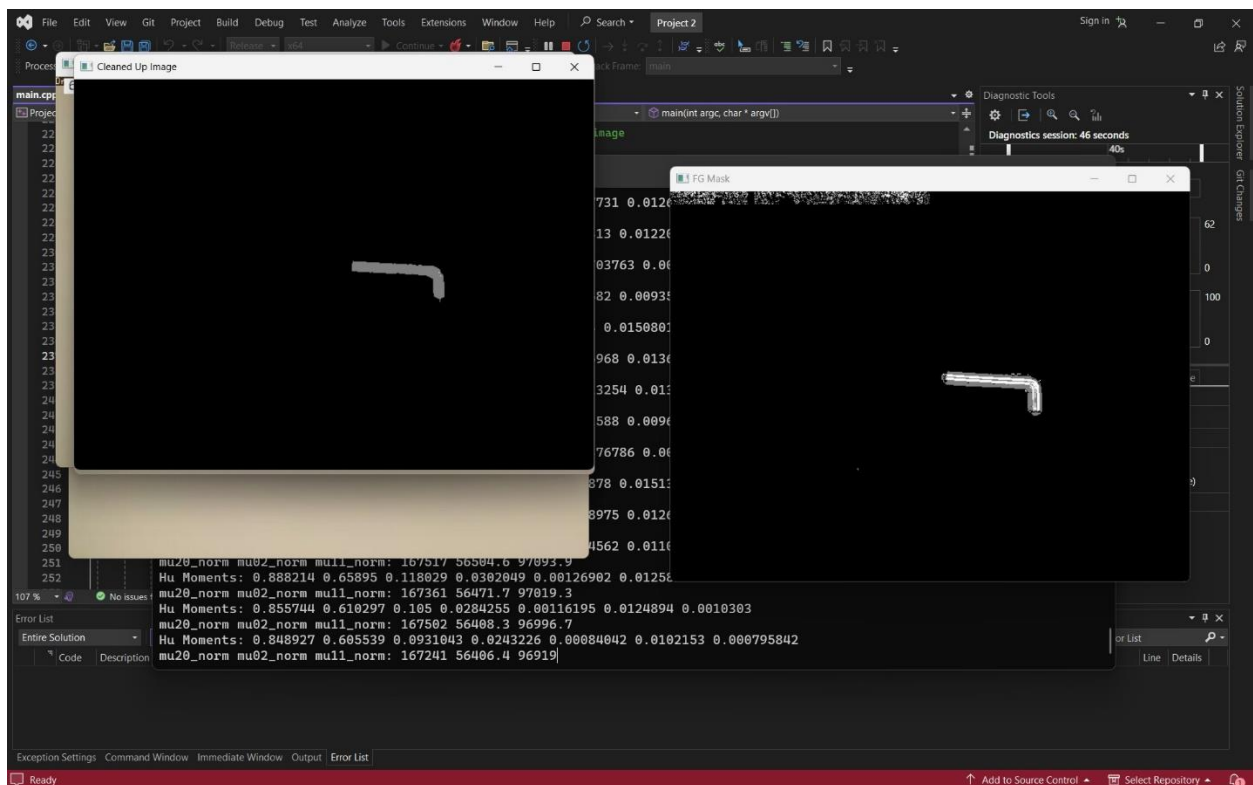
Alcm image of the Key



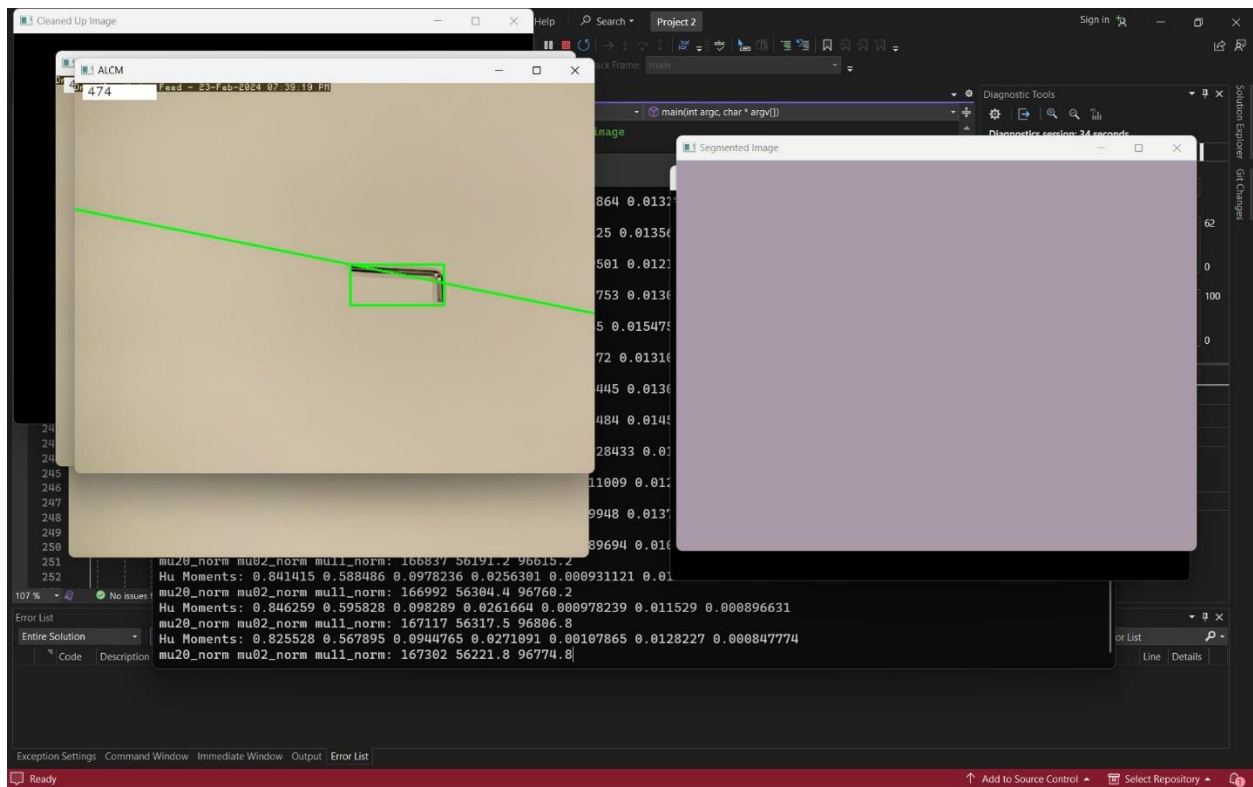
Original and threshold image of an Allen Key



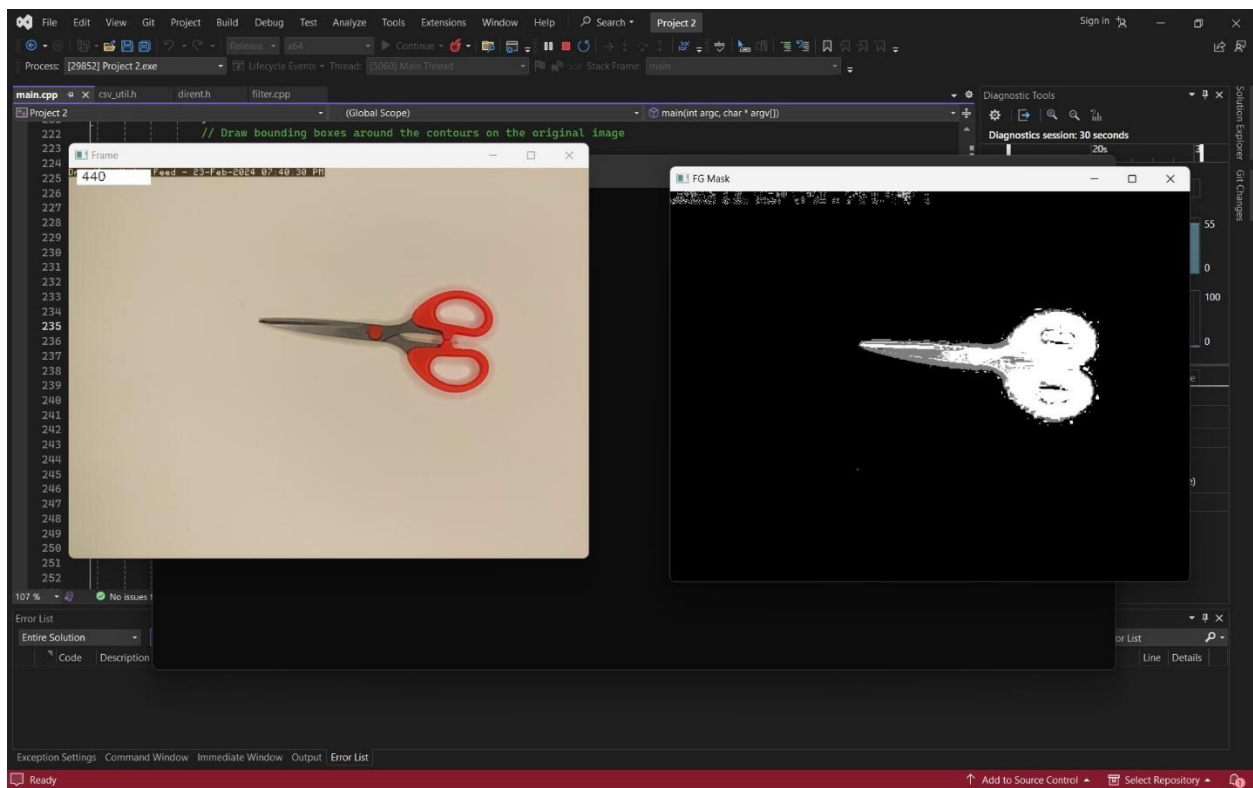
Cleaned up and threshold image of an Allen Key



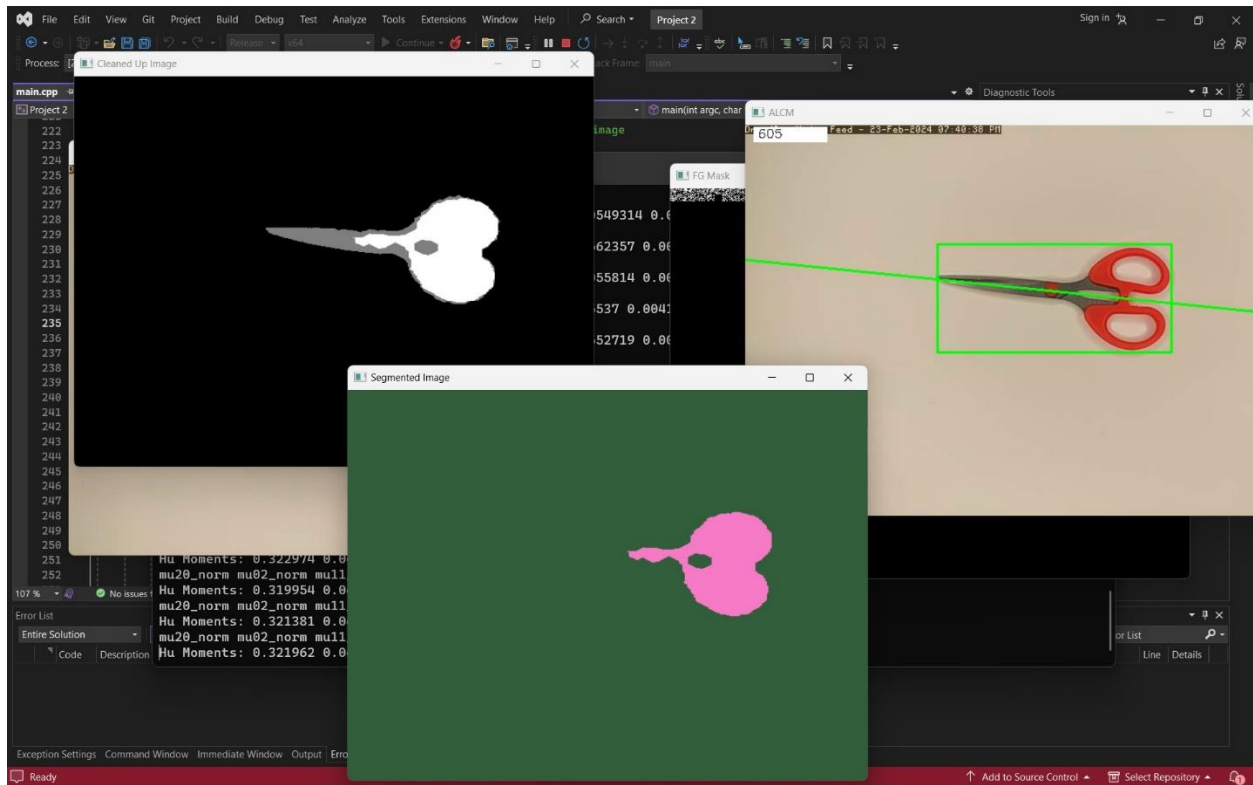
ALCM and segmentation of Allen Key



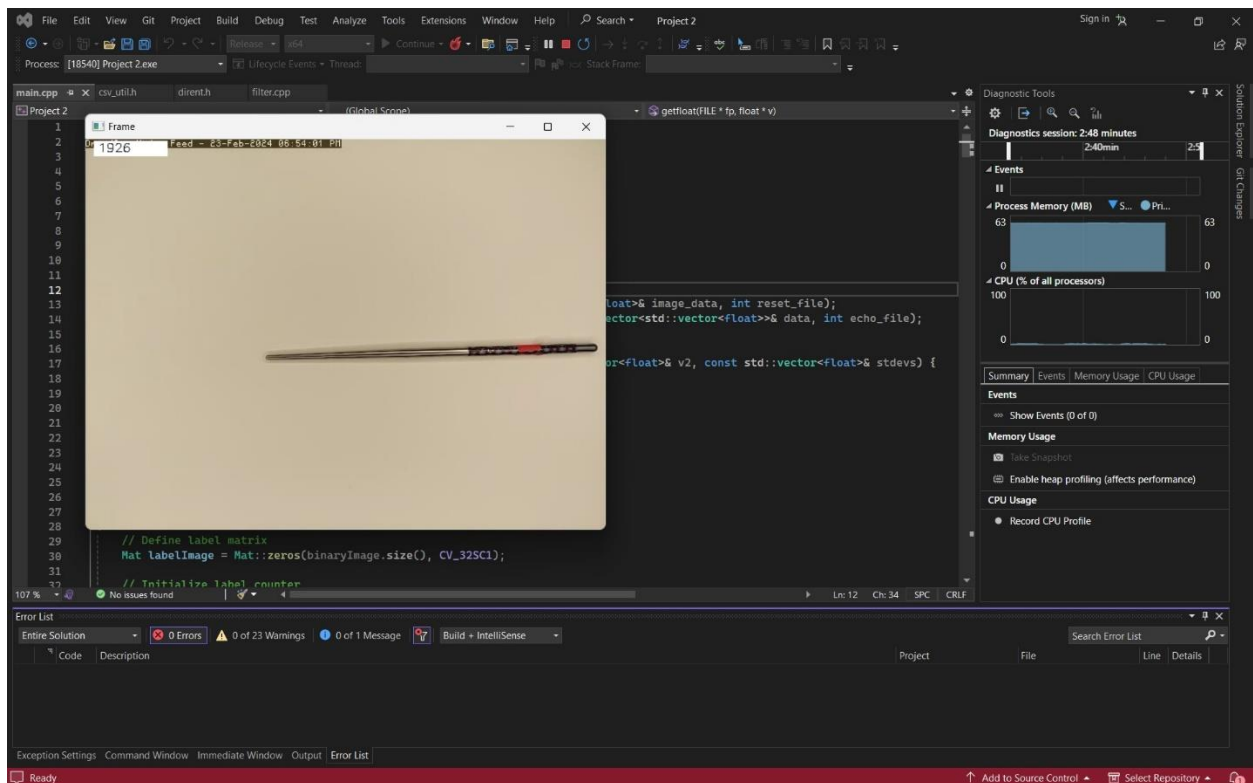
Original and threshold image of a Scissors



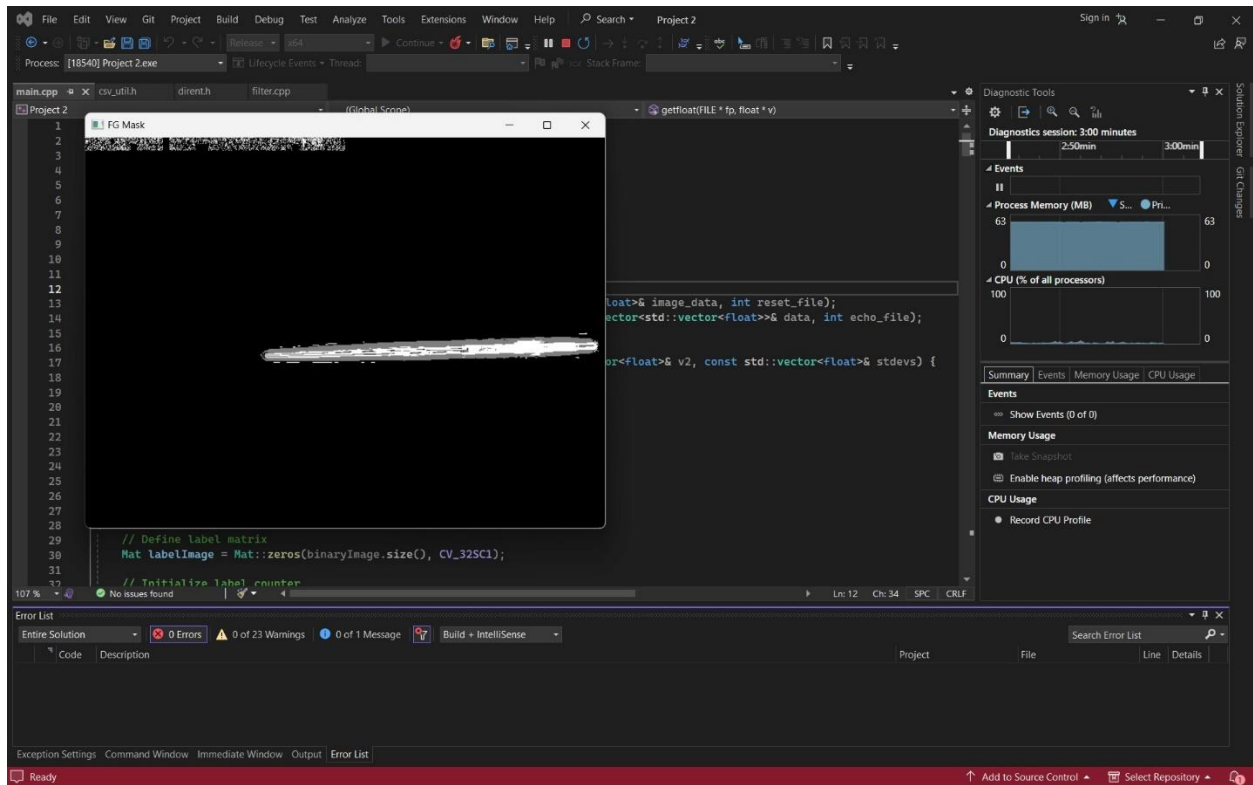
ALCM, cleaned up and segmentation of a scissors



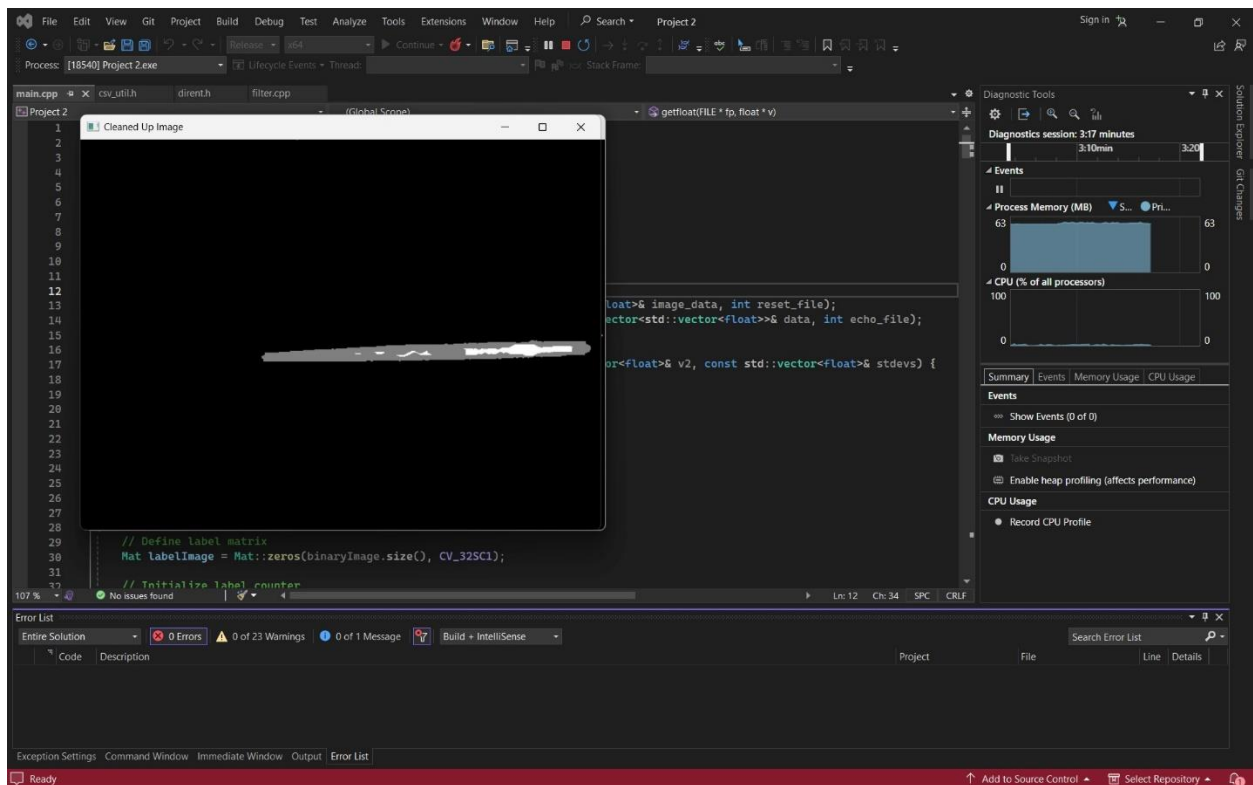
Original Image of a chopstick



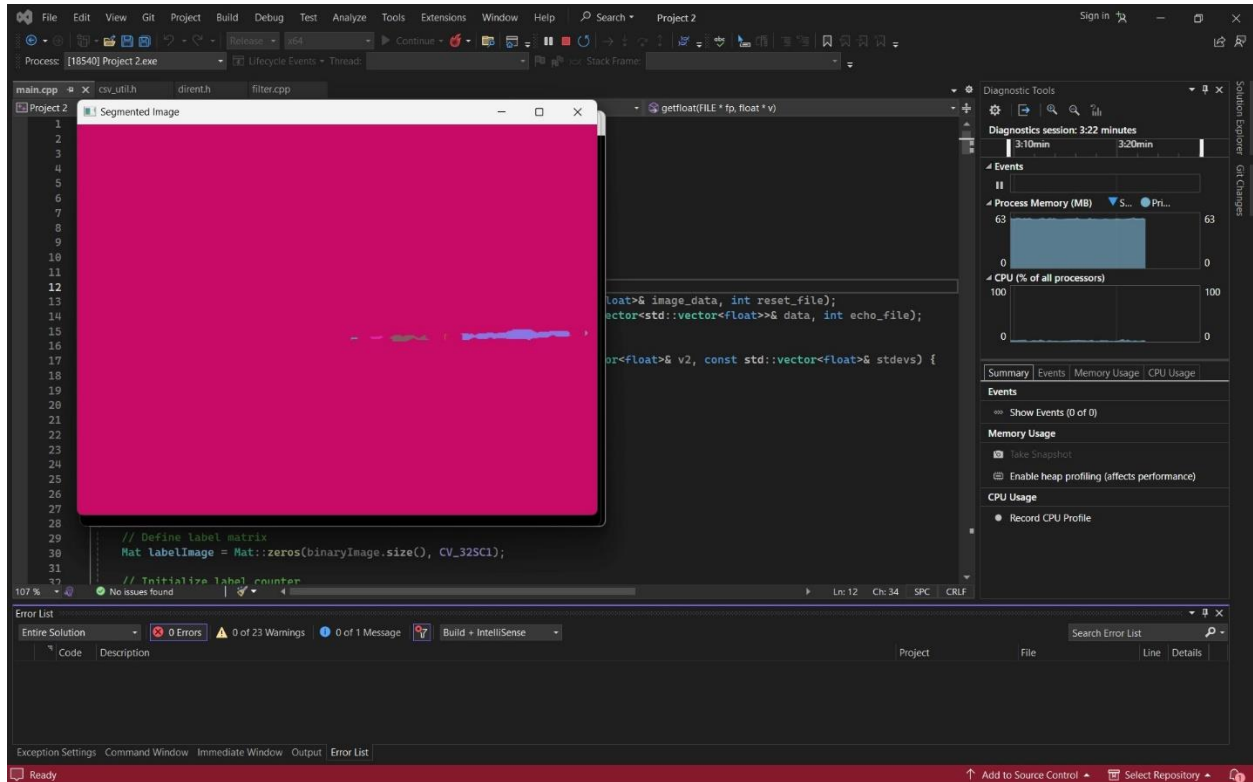
Threshold image of a chopstick



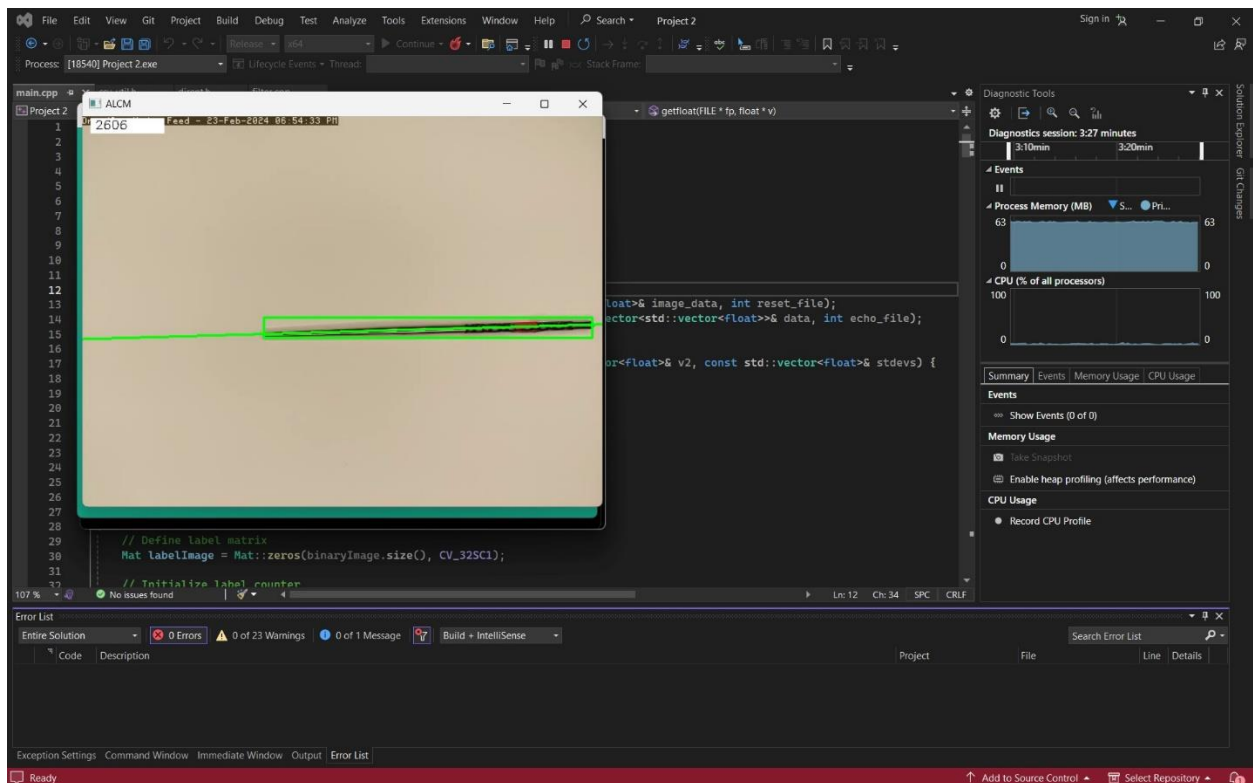
Cleaned up image of a chopstick



Segmented image of a chopstick



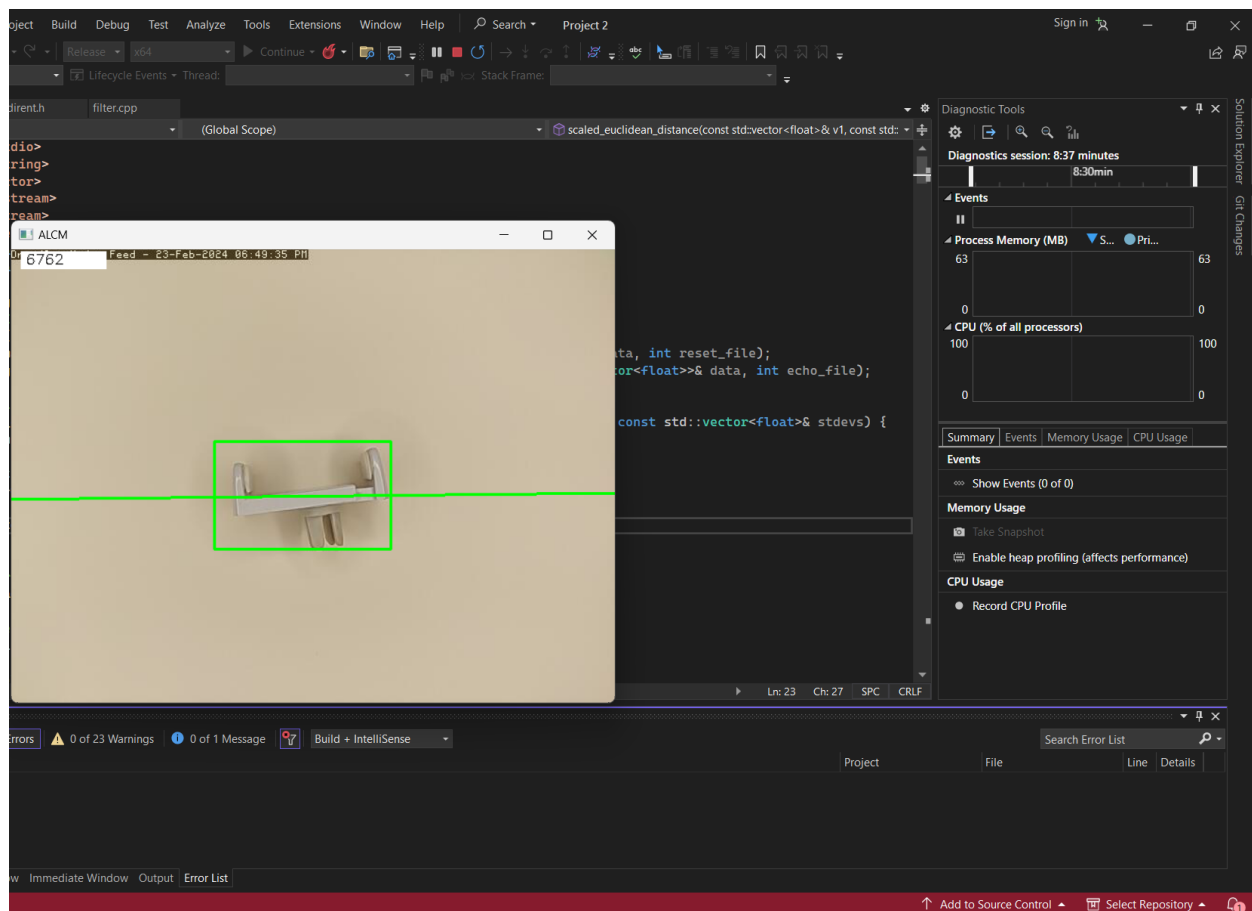
ALCM of a chopstick



Two other distance metrics were used other than the Euclidian distance metric they are the Cosine and Manhattan distances. While these metrics offer diverse perspectives Cosine distance focusing on the orientation rather than magnitude of vectors, and Manhattan distance emphasizing the sum of absolute differences. When implementing Cosine and Manhattan distances, we observed inconsistencies in accurately finding the Axis Least Central Moment (ALCM) for certain objects. This deviation could be attributed to how these metrics interpret spatial relationships and vector orientations differently from Euclidean distance. Cosine distance, commonly used in text analysis for its sensitivity to vector direction over magnitude, and Manhattan distance, preferred in grid-like or high-dimensional spaces for its robustness, introduced complexities in our context of spatial and shape analysis. These findings shows the criticality of choosing a distance metric that aligns with the specific characteristics and requirements of the data at hand, as well as the importance of thorough testing across various scenarios to ensure the reliability and accuracy of object recognition outcomes.

In the below image of a clamp we used the Manhattan distance and it showing a different ALCM compared to the original ALCM we got using the Euclidian distance

ALCM using Manhattan Distance



ALCM using Euclidean Distance

Fork classified as a spoon during classification ALCM and bounding box similar to a spoon



Acknowledgements & References:

- Computer Vision: Algorithms and Applications, 2nd ed.
- Object Detection Made Easy: Nicolai Nielson, YouTube Channel