Rucha Pendharkar 25th February 2024

Project III - Real-time 2-D Object Recognition

Introduction

The goal of this project is to have the system 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. To achieve this a sequence of tasks such as thresholding, cleaning, segmentation, feature calculation, is performed. Different methods of feature matching and classification are carried out using the database of objects. The results are analyzed and compared

Setup:

I used my phone to stream live video. The IP Webcam app proved to be very useful for this. I found a flat surface, in a well lit area and used white paper as a background. This helped me get better results overall.

Tasks:

1. Threshold the input video
2. Clean up the binary image
3. Segment the image into regions
4. Compute features for each major region
5. Collect training data
6. Classify new images
7. Evaluate the performance of your system
8. Capture a demo of your system working
9. Implement a second classification method

Task 1: Threshold the input video

A custom function was implemented for thresholding. The image is blurred using the 5x5 Gaussian Blur and converted to greyscale. Using a threshold value to compare, the function checks the intensity at each pixel and converts the background to black and foreground to white.

Figure 1: Input Video and Thresholded Video

Task 2: Clean up the binary image

After thresholding, there are some small halls and gaps, salt and pepper noise. I used erosion followed by dilation to clean up the image.

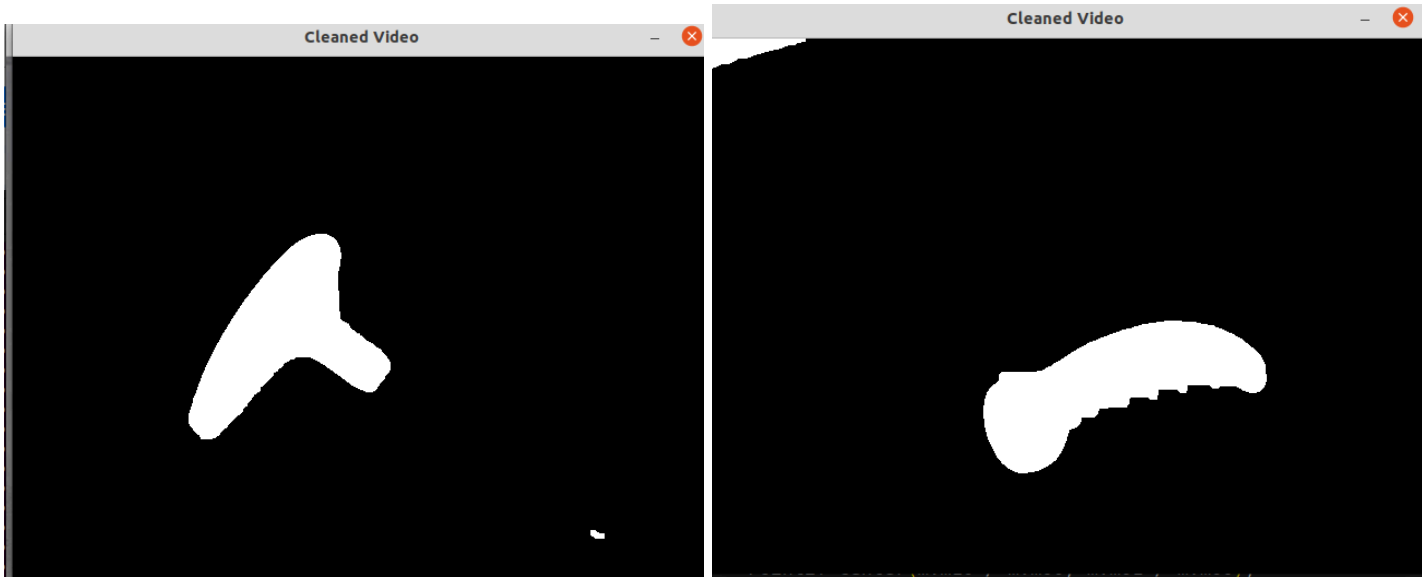

Figure 2: Cleaned Binary Image

Task 3: Segment the image into Regions

I used OpenCV's ConnectedComponentsWithStats function to calculate the region map. I am filtering out smaller areas. Total area and information regarding the bounding box is calculated. The end result is the image segmented into different regions (excluding the background)

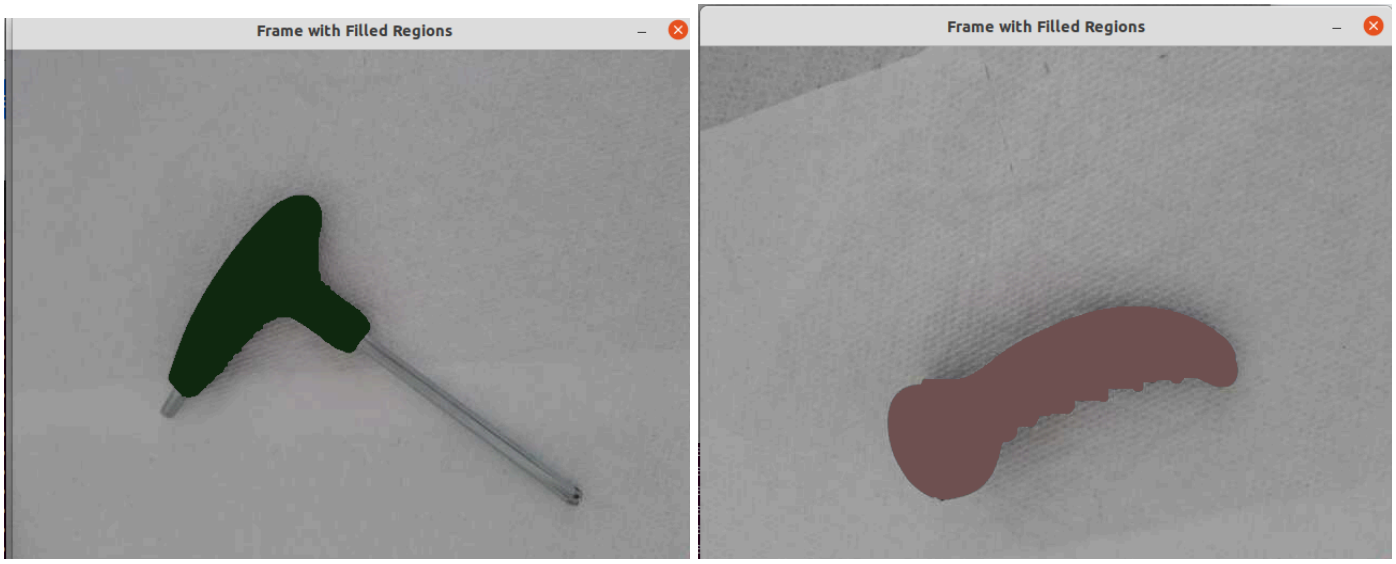

Figure 3: Segmented Video

Task 4: Compute Features for each major region

Using region map and region ID, different features were calculated.The bounding box was calculated to obtain some information regarding the boundary of the object.
Some features I calculated were percentFilled, ratio of the height and width of the bounding box, centroid of the object and Hu Moments

As translation invariant, I determined the centroid and central moments.
As translation, scale and rotation invariant, I determined percentFilled and Hu Moments
As translation and rotation invariant, I determined the ratio of the height and width of the bounding box,

I used the Moments function from OpenCV. For computing the center point, I used m10 and m01. central moments mu11, mu20, and mu02 were calculated for calculating the angle of orientation.

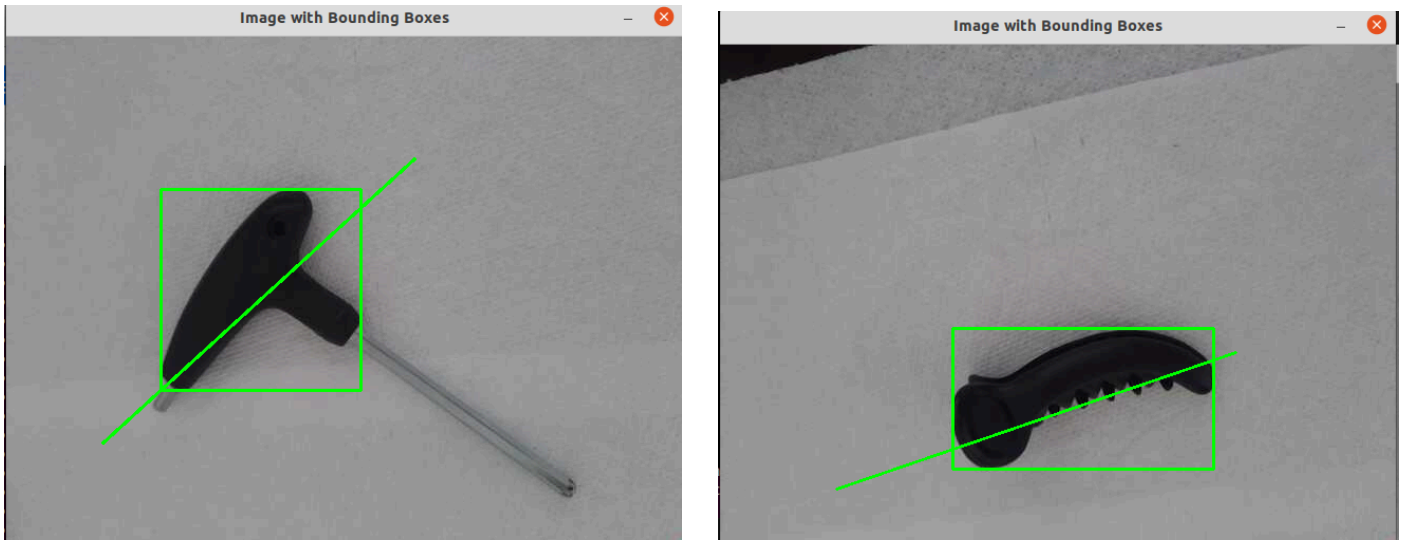

Figure 4: Objects with Bounding Boxes and Axis of least central moment

Task 5: Collect Training Data

If the user presses the key “n”, the program enters into Training Mode. Here the user is prompted to enter the label of the object being added to the database. The features are calculated from the live video. The features - center, Hu Moments, Percent Filled and the ratio of the height and width of the bounding box are all added to a CSV file

The CSV file is then later accessed to perform classification in subsequent tasks

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	898.654	176.566	0.534675	0.252565	0.0211021	0.0164379	0.000306145	0.00826097	1.38512E-06	0.562636	0.247263	hairbrush	
2	695.125	520.398	0.575565	0.297747	0.0208225	0.0165632	0.000307587	0.00903685	2.45039E-06	0.236691	0.22487	hairbrush	
3	475.327	71.809	1.01189	0.993884	0.0236604	0.0212659	0.00047701	0.021148	3.20105E-06	0.415967	0.105295	pen	
4	100.897	53.3498	0.227258	0.00525976	0.000206329	0.000291281	-1.97929E-08	-1.94843E-05	-6.86102E-08	0.579292	0.837079	pen	
5	106.263	59.6833	0.408154	0.00721154	0.0101367	0.00505561	3.01013E-05	-4.09404E-05	2.00934E-05	0.332366	0.872483	pen	
6	240.165	184.444	0.349189	0.0844938	0.0035943	0.000876903	9.16158E-07	6.70429E-05	-1.25869E-06	0.379182	0.378861	clip	
7	31.0453	9.93329	0.283936	0.0380582	0.00387094	0.000968639	1.06013E-06	4.88553E-05	1.54731E-06	0.689089	0.423077	clip	
8	345.955	184.856	0.346552	0.0811292	0.00304366	0.000755889	6.91328E-07	8.34904E-05	9.14655E-07	0.39493	0.423765	clip	
9	218.884	68.9161	0.545652	0.263409	0.00254332	0.000457832	2.19113E-07	-1.76393E-05	4.42789E-07	0.366166	0.218102	lighter	
10	195.112	38.4103	0.483799	0.201908	0.00353705	0.00199543	5.26281E-06	0.000748963	6.37069E-07	0.639687	0.217184	lighter	
11	153.402	115.75	0.417529	0.1402	0.00888188	0.00577455	4.12424E-05	0.00203474	3.05174E-06	0.277524	0.225868	lighter	
12	44.3411	170.032	0.383312	0.110705	0.00793642	0.00401703	2.24697E-05	0.00125505	3.09137E-06	0.651574	0.224391	lighter	
13	111.928	72.1198	0.249824	0.0137798	0.00712875	0.000459516	1.58008E-07	4.10697E-05	-8.16535E-07	0.524799	0.772727	allenkey	
14	121.718	98.0856	0.243075	0.0110863	0.00641768	0.000327143	-1.54241E-08	2.47056E-05	-4.73768E-07	0.497021	0.817459	allenkey	
15	118.737	123.931	0.261785	0.0133326	0.00868454	0.000485487	-3.14082E-07	2.81551E-05	-9.46103E-07	0.459076	0.787151	allenkey	
16	289.076	168.131	0.563981	0.285706	0.0212324	0.0169824	0.000322476	0.00907724	7.84639E-07	0.267019	0.232919	hairbrush	
17	87.5409	149.713	0.94522	0.861698	0.0486513	0.0438061	0.00202229	0.0406306	9.46271E-06	0.204449	0.121437	pen	
18	116.666	151.516	0.99765	0.965943	0.00781468	0.00680125	4.958E-05	0.00664708	5.97572E-07	0.158481	0.104384	pen	
19	100.619	100.21	0.251598	0.0124275	0.00800758	0.000700486	2.8298E-07	5.6243E-05	1.6347E-06	0.437874	0.785172	allenkey	
20	81.9474	87.1808	0.252023	0.0148825	0.00729779	0.000542159	3.38716E-07	5.30254E-05	1.02384E-06	0.493355	0.762819	allenkey	
21	104.585	133.562	0.340904	0.0778907	0.00404384	0.00137742	3.10231E-06	0.000358693	-9.71393E-07	0.357535	0.418397	clip	
22													
23													
24													

Figure 5: CSV File

Task 6: Classify new images

The below figures are results from using Nearest Neighbor. Here I used scaled euclidean as a distance metric. I calculated the standard deviation for each feature using the training dataset I had. This gave me pretty decent results

Figure 6: Classified Images

Task 7: Evaluate the performance of your system

For evaluating the performance of my system, I examined five inputs of each object. The below table populates the results of the experiment. Majority of the classifications can be observed along the diagonal, indicating a fairly good classification

Predicted\Actual	Pen	Clip	Hairbrush	Allen Key	Lighter
Pen	4	0	0	0	0
Clip	0	3	1	0	1
Hairbrush	1	1	4	0	0
Allen Key	0	1	0	5	1
Lighter	0	0	0	0	3

Table 1: Confusion Matrix using Nearest Neighbor

Task 8: Video

Video of the system classifying objects
https://drive.google.com/file/d/1wbQee34rnDJd40o2KvI0uNmRVyO_nRxi/view?usp=sharing

Task 9: Implementing K-nearest neighbors

I expanded my database to include more images of the objects in various orientations.

For implementing K - nearest neighbors I found the closest K-examples in each class. I then summed the distances within each class. The unknown vector can be labeled corresponding to the smallest sum.

Predicted\Actual	Pen	Clip	Hairbrush	Allen Key	Lighter
Pen	4	0	0	0	0
Clip	0	5	1	0	1
Hairbrush	1	0	4	0	0
Allen Key	0	0	0	5	0
Lighter	0	0	0	0	4

Table 2: Confusion Matrix using K-Nearest Neighbor

Overall, I found K-nearest neighbors yields better results than just nearest neighbors. There are fewer instances of misclassifications

Extensions

For the extension I added the capability to **recognize more objects** to my database, and was able to detect them pretty accurately

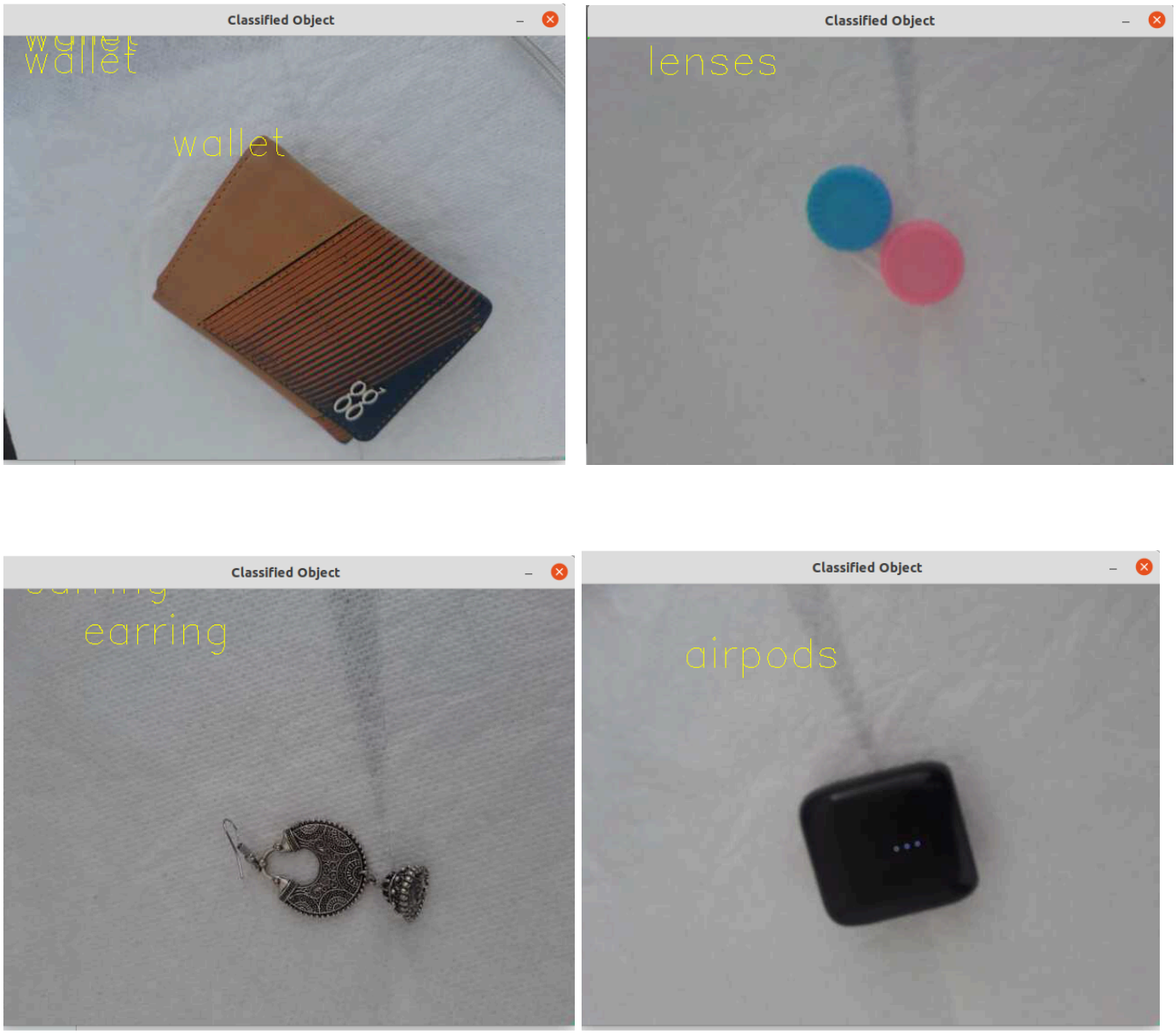

Figure 7: More Objects, classified

Actual\Predicated	Pen	Clip	Hairbrush	Allen Key	Lighter	wallet	Lenses	earring	airpods
Pen	9	0	0	0	0	0	0	0	0
Clip	0	9	2	0	1	0	0	0	0
Hairbrush	1	0	7	0	0	0	0	0	0
Allen Key	0	0	0	9	0	0	0	0	0
Lighter	0	0	0	0	8	0	0	0	0
wallet	0	0	0	0	0	7	0	0	2
lenses	0	0	0	0	0	0	9	1	0
earring	0	0	0	0	0	0	0	8	0
airpods	0	0	0	0	0	2	0	0	7

Table 3: Confusion Matrix for larger database of objects

Reflection

Albeit challenging, this project was really fun. Like Project I, it was really important to get a working pipeline ready. I feel the structure of the tasks made this really easy.

I learnt about OpenCV functions and handling different data types efficiently. I learnt a lot about feature calculation and moment calculation and my understanding of feature matching also deepend. I also understood the importance of lighting and shadows. It took me a while to eliminate these environmental errors.

Acknowledgment

I would like to thank Prof Bruce Maxwell for his notes and class materials. His help in debugging my feature function was invaluable, and fixing my axis of least central moments. I am grateful for TA's Erica Shephard and Tejaswini Deore help as well to help me fix various parts of my code.

References

1. CS5330 Notes and Lecture Materials
2. OpenCV Documentation- https://docs.opencv.org/3.4/d8/d23/classcv_1_1Moments.html#a1f437e60fd477d2ea4b5a3f0c2b8a9d4
3. https://www.geeksforgeeks.org/connect-your-android-phone-camera-to-opencv-python/