Project 3: Real-time 2-D Object Recognition

Student: Yunyi Chi

Description:

In this project, I developed a Real-time 2-D Object Recognition system. The system works by performing a series of image processing operations on the input video stream, including thresholding, morphological processing, and segmentation. Next, the system computes a set of features that can uniquely describe each object. Using these features, I trained our own database of objects.

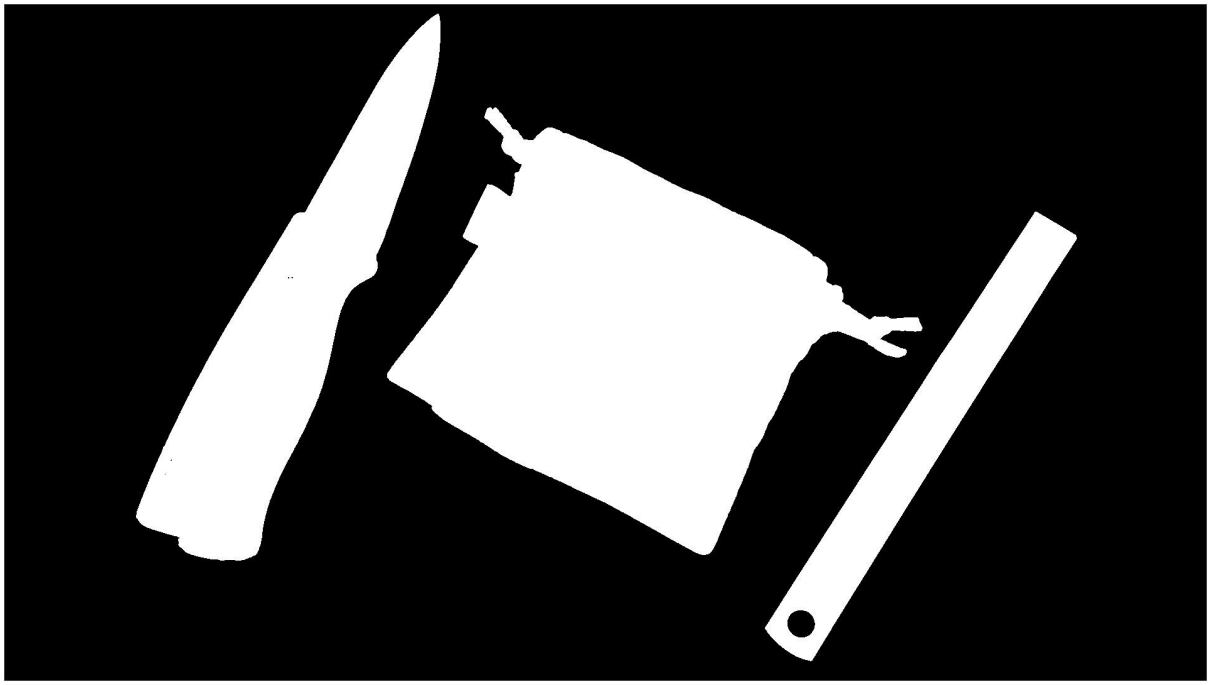
To classify new objects, I used the Euclidean distance metric and K-Nearest Neighbor algorithm. Specifically, I computed the distance between the feature vector of the unknown object and the feature vectors in the database, and assigned the object to the class of the closest neighbor.

Finally, I tested the system's performance using a dataset of known objects and achieved high accuracy. Besides, I have 3 extensions for this project: 1. I improved my system so that it can simultaneously recognize three objects, resulting in a 300% increase in efficiency. 2. I add more than the required ten objects to the DB and make my system able to recognize more objects. 3. Write a new mode, which can call a new classify_image_manhattan_distance() method to make recognition. This method uses a new Manhattan distance matrix for comparing feature vectors.

Task 1: Threshold the input video (write the code from scratch)

For this task, I wrote the code from scratch.
first I use two 1x5 separable gaussian filters to blur the input video frame, it can make the regions more uniform.
Then, I transfer the input video frame to a greyscale video frame.
Finally, I threshold the greyscale frame to make a binary image, all areas with a greyscale value bigger than 140 should be background, The rest is the foreground.

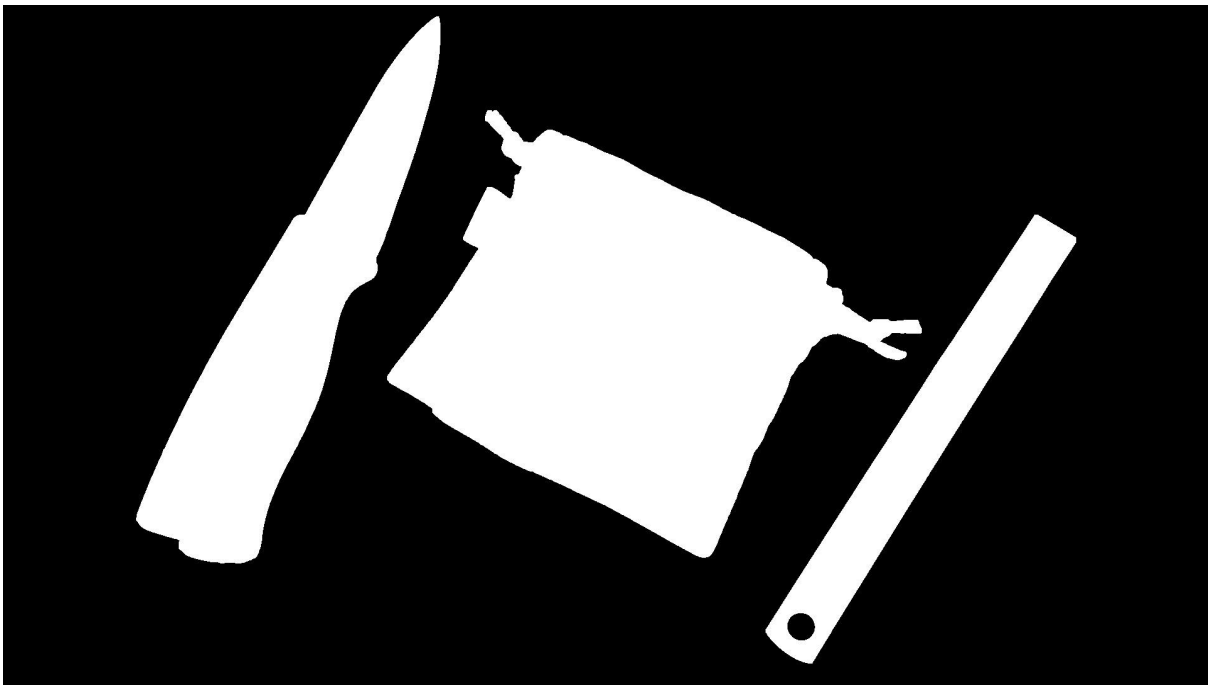
original frame (include 3 objects)



binary image (include 3 objects)

Task 2: Clean up the binary image

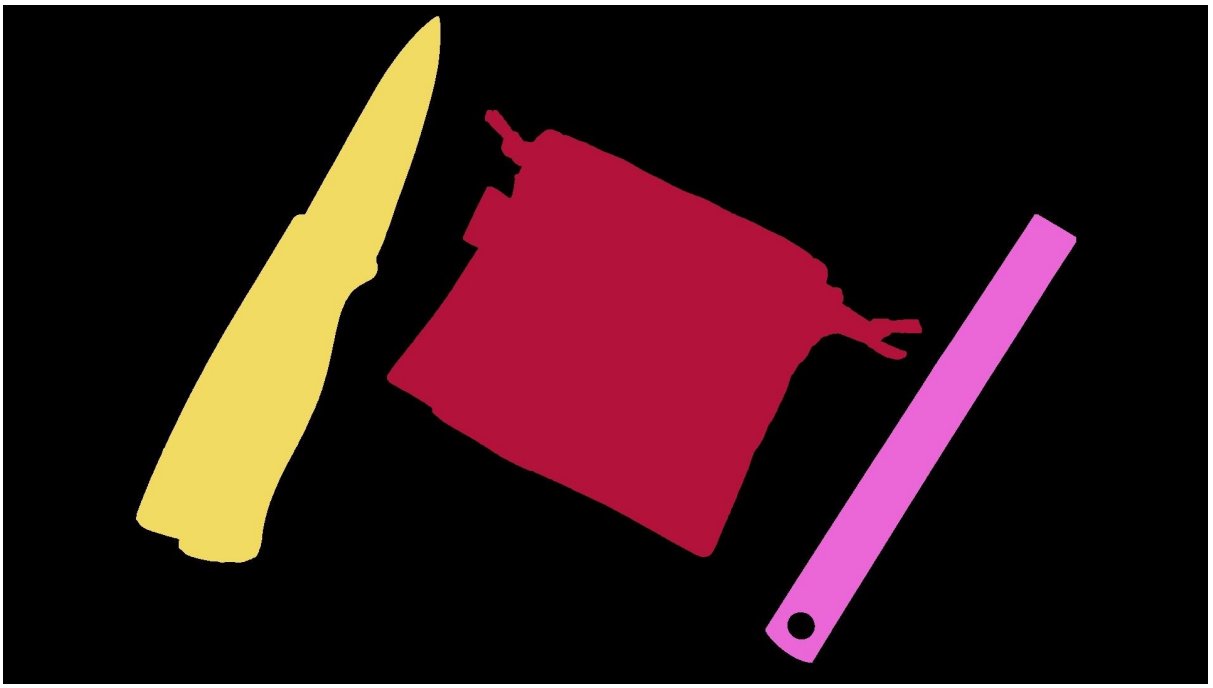
For this part, I use the openCV built-in methods cv::morphologyEx() and cv::getStructuringElement() to achieve morphological_filtering, because cv::morphologyEx() includes erosions and dilations process, so it works very well. The result is shown as below:



clean binary image (include 3 objects)

Task 3: Segment the image into regions

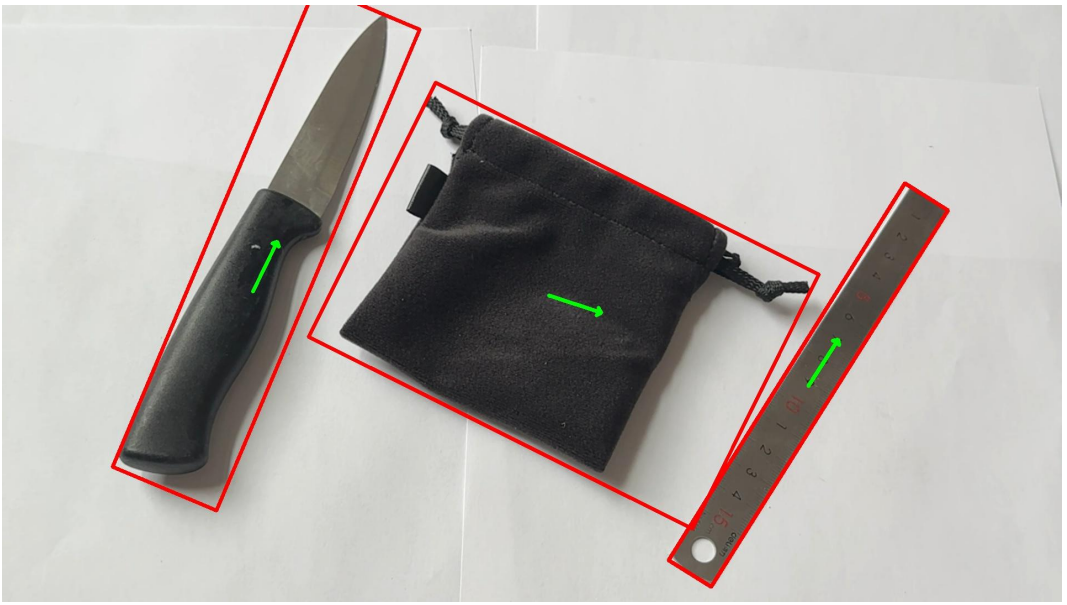
For this task, I use the openCV built-in method cv::connectedComponentsWithStats(), I sorted the return regions based on the area. All regions less than 4000 pixels should be ignored. Finally, I colored the biggest 3 connected regions, set black to all the other regions.



original

Task 4: Compute features for each major region

For this part, first I use cv::moment() function to get a set of moments, This function can return spatial moments, central moments and central normalized moments for object.
then I calculate the axis of least central moment and the oriented bounding box, also draw them in the original frame.
Finally, I calculate the cv::HuMoment(), this function calculates seven Hu invariants, these values are proved to be invariants to the image scale, rotation, and reflection except the seventh one, whose sign is changed by reflection. As a result, HuMoment is more useful to recognize objects.



frame showing the axis of least central moment and the oriented bounding box

Task 5: Collect training data

On the command line, if you set the second argument to be “train”, then you will start the program with a training mode. Under training mode, you can input ‘n’ to get the feature of the current object and save it to temp database, if you input ‘s’ , you will write the newest feature database into a csv file. Some data are shown below.

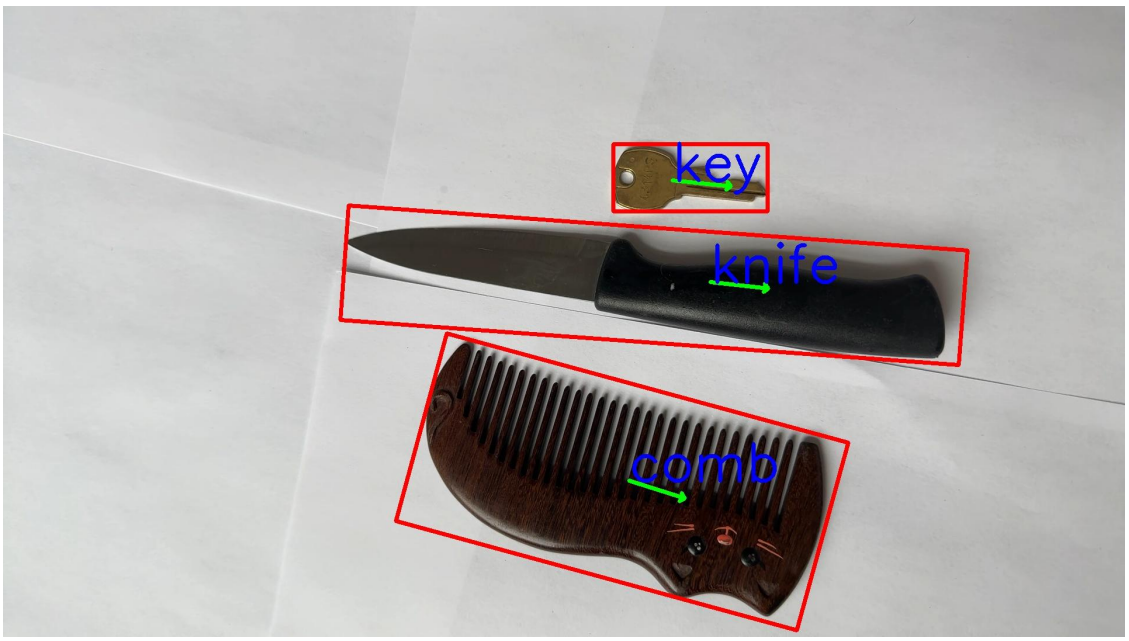
toothbrush	0.900585	0.774007	0.0317393	0.0316983	0.00100542	0.0277841	-5.41674E-06
toothbrush	0.87878	0.734896	0.0184557	0.0191085	0.000358839	0.0162969	-1.85952E-06
toothbrush	0.842421	0.672985	0.0149412	0.0154364	0.000234425	0.0125612	-1.15642E-06
scissor	0.497774	0.181951	0.0298264	0.00927011	0.000151503	0.00360067	2.84145E-05
scissor	0.526061	0.197966	0.0388707	0.0110643	0.000225777	0.00448904	4.09105E-05
scissor	0.488848	0.173214	0.0296032	0.00903397	0.000144946	0.00340312	2.85774E-05
comb	0.237862	0.0288622	0.000224047	2.09577E-05	-1.1329E-09	-2.94702E-06	-8.82564E-10
comb	0.237503	0.0287746	0.000228225	2.08012E-05	-1.19888E-09	-3.04177E-06	-7.85389E-10
comb	0.238255	0.0290598	0.000258493	2.33912E-05	-1.78866E-09	-3.91488E-06	-3.30137E-10
knife	0.50547	0.224628	0.0164298	0.0123062	0.000174985	0.00583232	7.45049E-07
knife	0.52346	0.241288	0.0289462	0.0226709	0.000580754	0.0111358	2.89327E-06
knife	0.512131	0.231704	0.0166783	0.0126279	0.000183261	0.00607322	-3.41007E-07
key	0.287963	0.0441053	0.0107869	0.00432043	2.94443E-05	0.000900477	-1.71933E-06
key	0.28995	0.0460905	0.0103003	0.00425468	2.81395E-05	0.000909551	-1.22147E-06
key	0.287152	0.0438508	0.0105318	0.0041774	2.7658E-05	0.000867677	-1.6693E-06
wallet	0.20514	0.0148459	9.56922E-05	1.60951E-05	6.27592E-10	1.95441E-06	-7.14986E-11
wallet	0.205323	0.0148839	7.65268E-05	1.46286E-05	4.88985E-10	1.73509E-06	2.13839E-11
wallet	0.203121	0.0140687	7.55535E-05	1.20083E-05	3.57462E-10	1.41493E-06	-5.52243E-11
cake	0.161387	0.0006497	3.93576E-06	2.50729E-08	-7.10914E-15	-6.37374E-10	3.3906E-15
cake	0.161786	0.000785026	1.39159E-06	2.70784E-09	-1.46322E-16	-2.92069E-11	-7.88652E-17
cake	0.162109	0.000876363	3.62122E-06	1.60996E-08	-3.22085E-15	-2.1931E-10	-2.17655E-15

training data

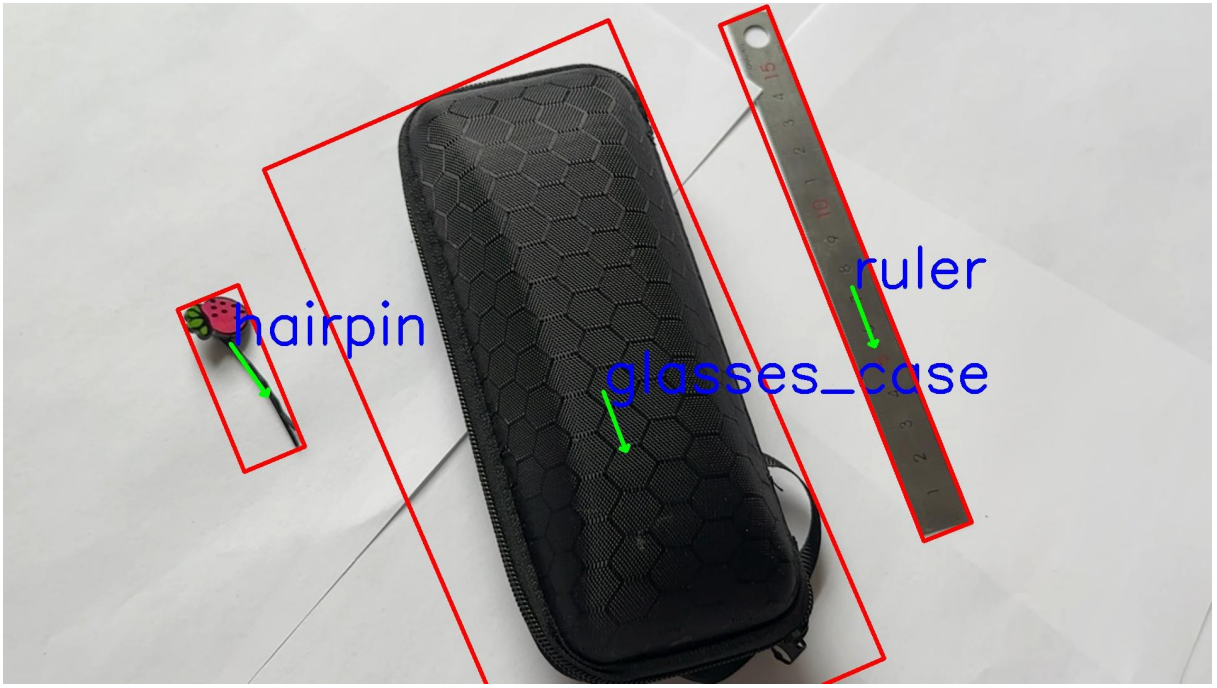
Task 6: Classify new images

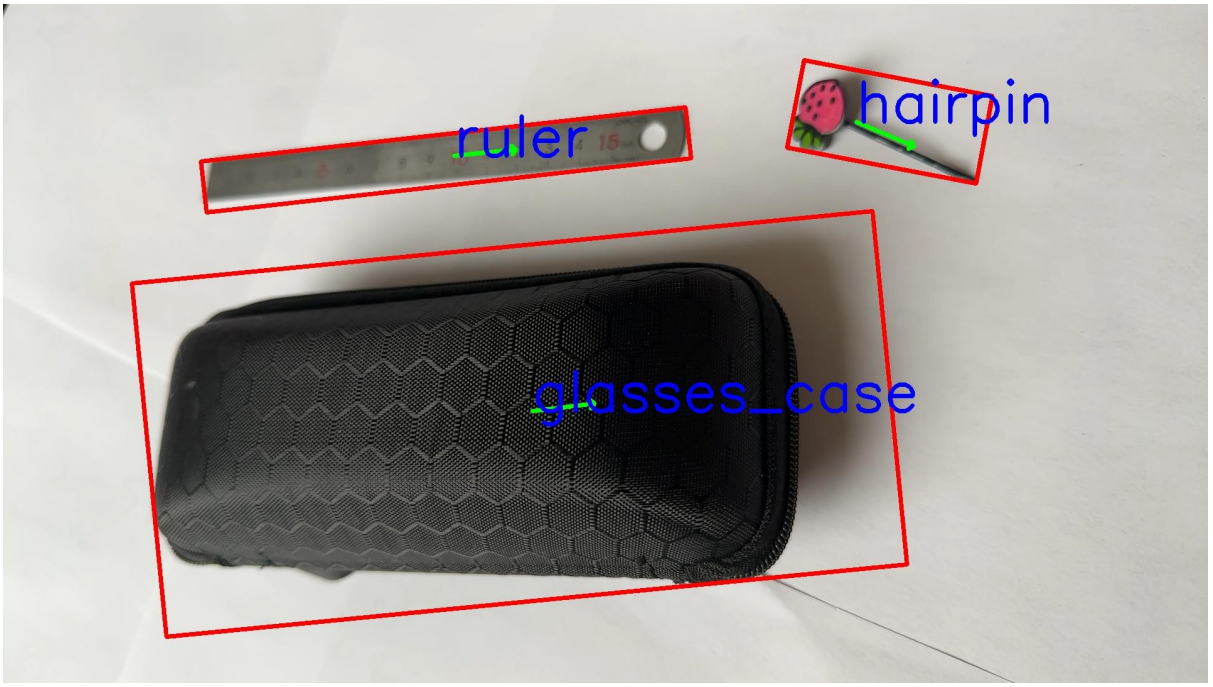
For this task, for every object, I calculate the distance between it's feature vector and the feature vectors in the database with a scaled Euclidean distance metric, then I use nearest-neighbor recognition to label the unknown object according to the closest matching feature vector in the object DB, and text the label on the object frame. To start a nearest-neighbor recognition, please set the second argument to 'nn' in command line.

I conducted a comprehensive testing process on the objects in the video, which included evaluating two different shapes, angles, and scaling variants for each object, as shown in the image below. This rigorous testing approach ensured that the system could accurately detect and classify objects under various conditions, making it suitable for real-world applications.

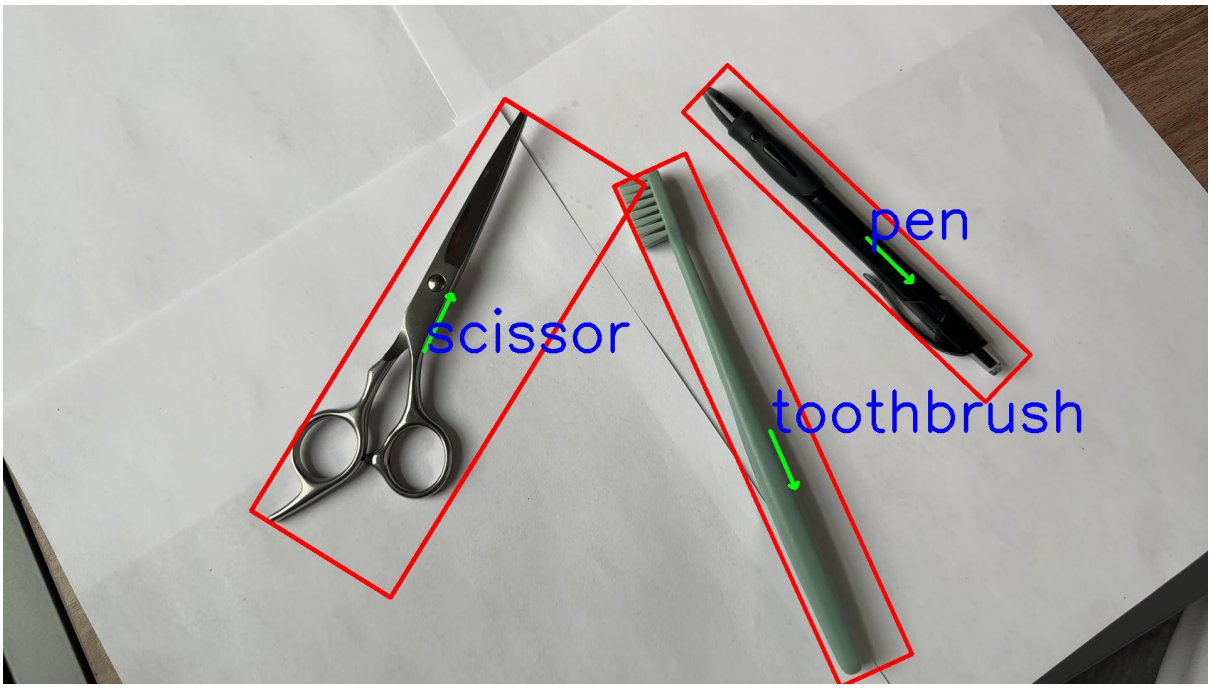

key
knife
comb

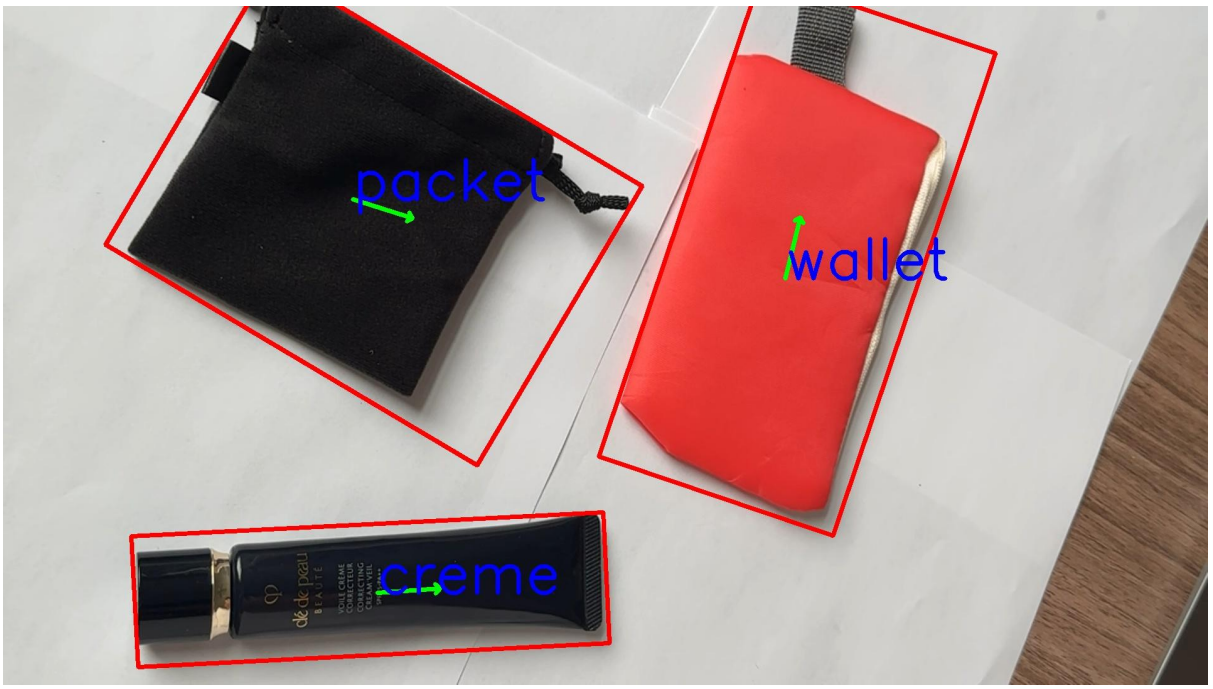

comb
knife
key


hairpin
glasses_case
ruler


ruler
hairpin
glasses_case


scissor
toothbrush
pen


pen
scissor
toothbrush


packet
wallet
creme


packet
wallet
creme

Task 7: Implement a different classifier (write the code from scratch)

On the command line, if you set the second argument to be “knn”, then you will start the program with a K-Nearest Neighbor recognition mode. The specific code can be seen in the knn_classify() method in processing.cpp.

Task 8: Evaluate the performance of your system

After training the required data, I performed 5 recognitions on each object, and the final confusion matrix is shown in the figure below.
The rows and columns representing the actual and predicted labels.

	pen	toothbrush	scissor	comb	knife	key	wallet	ruler	packet	hairpin	creme	glasses_case
pen	4	1										
toothbrush		4						1				
scissor			5									
comb				5								
knife					5							
key						5						
wallet							4		1			
ruler		1						4				
packet							1		4			
hairpin										5		
creme											5	
glasses_case												5

Task 9: Capture a demo of your system working

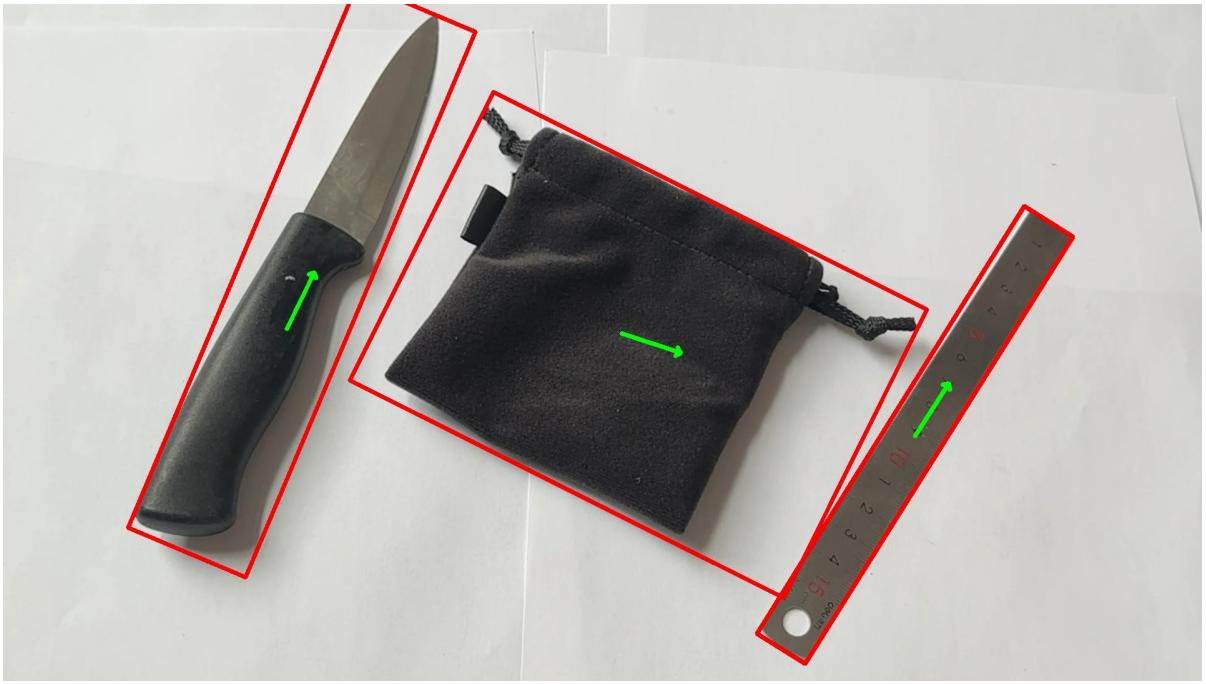
The link for the video is shown below:
https://drive.google.com/file/d/1pHVdz_cX1X8GEoREpUqbTlO1LZxW012Z/view?usp=sharing
it includes the instruction for starting ‘nn’ mode, ‘knn’ mode and ‘train’ mode.

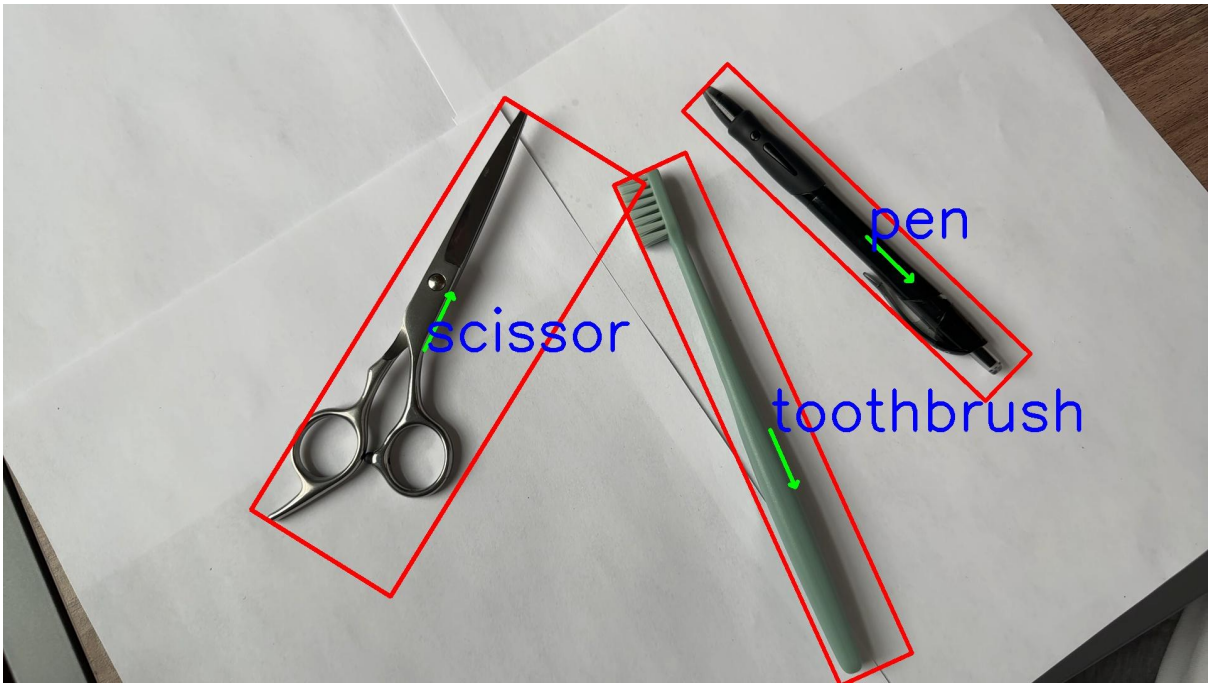
Extensions (3 extensions)

- **Enable recognition of 3 objects simultaneously**
- **Recognize more than 10 objects**
- **Apply a new distance metrics for comparing feature vectors**

1. Enable recognition of 3 objects simultaneously.

In task3, after getting connected regions in the binary image, I also sort the regions by their area, and at the same time retain the 3 largest regions. The three regions can also be used to compute features, collect training data and make classification at the same time, so it is resulting in a 300% increase in training and classification efficiency.





2. Recognize more than 10 objects

From the data.csv in the uploading video, you can see there are training data of 13 kinds of objects. I show the classification of 12 of them, you can see them in task 6.

3. Apply a new distance metrics for comparing feature vectors

   On the command line, if you set the second argument to be “manhattan”, then you will start the program with a new recognition mode. It calls a new classify_image_manhattan_distance method to make recognition. This method uses a new Manhattan distance matrix instead of Euclidean distance for comparing feature vectors, which is good as well.

What I learned

From this project, I learn the following things

1. Learn how to make a binary image from scratch.
2. Learn how to do some morphological filtering to remove the dot and hole in the binary image.
3. how to get connected regions from a binary image in opencv
4. Learn the different features of objects, and some of them are not affected by the angles, and scaling.
5. Write K-Nearest Neighbor algorithm from scratch.

Acknowledgement

Maxwell's lecture notes, by *Dr. Bruce A. Maxwell*

Computer Vision: Algorithms and Applications, 2nd ed, by *Richard Szeliski*

OpenCV Tutorials: https://docs.opencv.org/4.x/d9/df8/tutorial_root.html