NAME	: JOSE JOSEPH THANDAPRAL
NUID NO.	: 002102407
COURSE CODE	: CS 5330
COURSE NAME	: PATTERN RECOGNITION AND COMPUTER VISION
CRN	: 36458
FACULTY	: PROF. BRUCE MAXWELL

NORTHEASTERN UNIVERSITY, BOSTON

PROJECT 3

OBJECTIVE

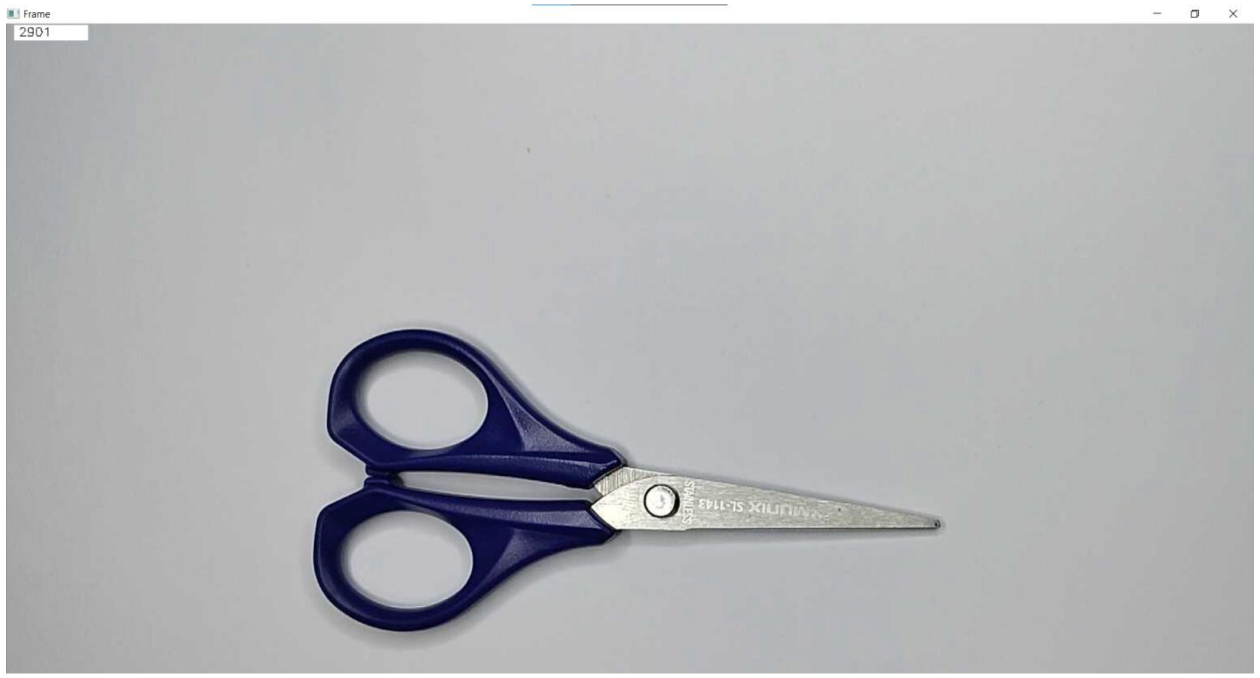
The report includes the results and observations of the third project involving using the OpenCV package to perform 2-D real-time object recognition. The implemented system uses a phone camera to get image frames in real-time and computes the rotation, scale and translation invariant features of the object after thresholding to differentiate the foreground object of interest from the background, essentially giving us a cleaned up segmented binary image of the object. The given implementation has an explicitly defined function for the 2 pass algorithm for segmentation whereas the rest of the implementation uses OpenCV library function. The user is prompted to enter inputs to go into a Training or Classification mode where it can either compute features for new object labels and add to the .csv database file or it can classify the objects in real-time. This is done in a recurring while loop until the project file is stopped. There is also a K-NN implementation of the distance matching as compared to the initial scaled Euclidean distance to calculate the top match for object for classisification. Further, we manually calculate the confusion matrix to examine the performance of the implemented system. In addition to actually objects, additional objects in the database file include paper cut outs of objects to test the system further.

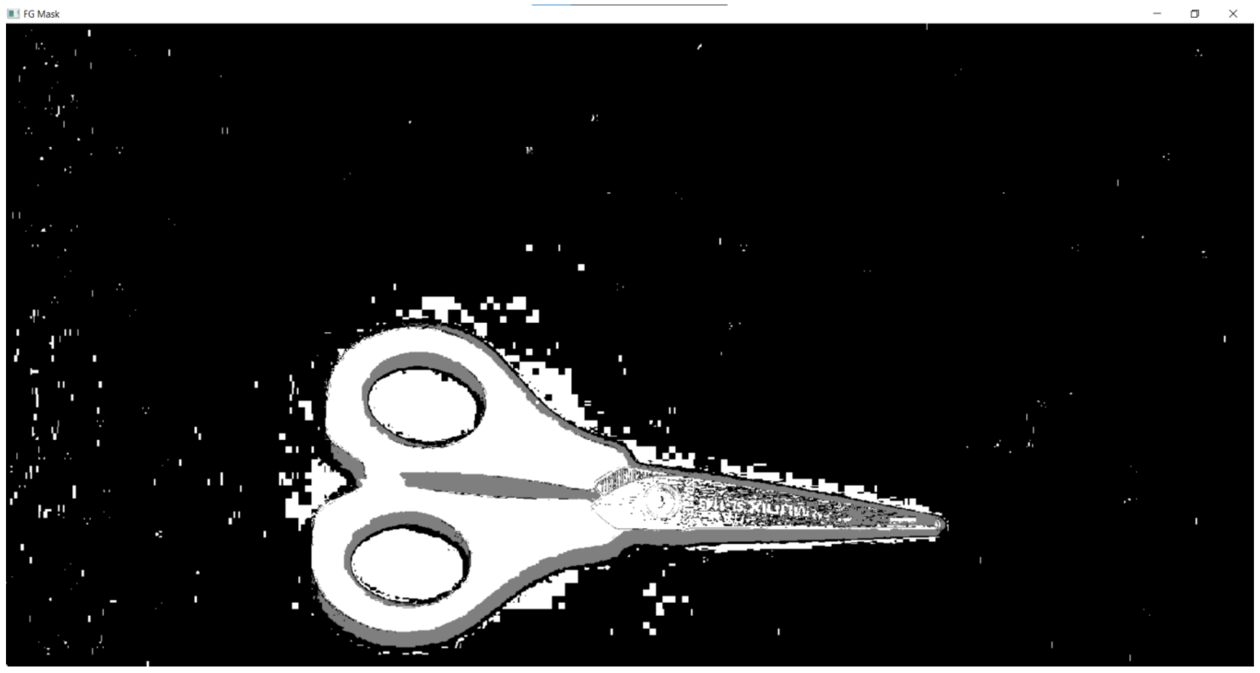
NOTE: The implemented code for the .csv databse file creation ,traversal and manipulation do not include a .h file as the functions are defined in csv_util.cpp inside the same Visual Studio Code project solution. The function prototypes declared at the top of main.cpp call the required functions from csv_util.cpp.

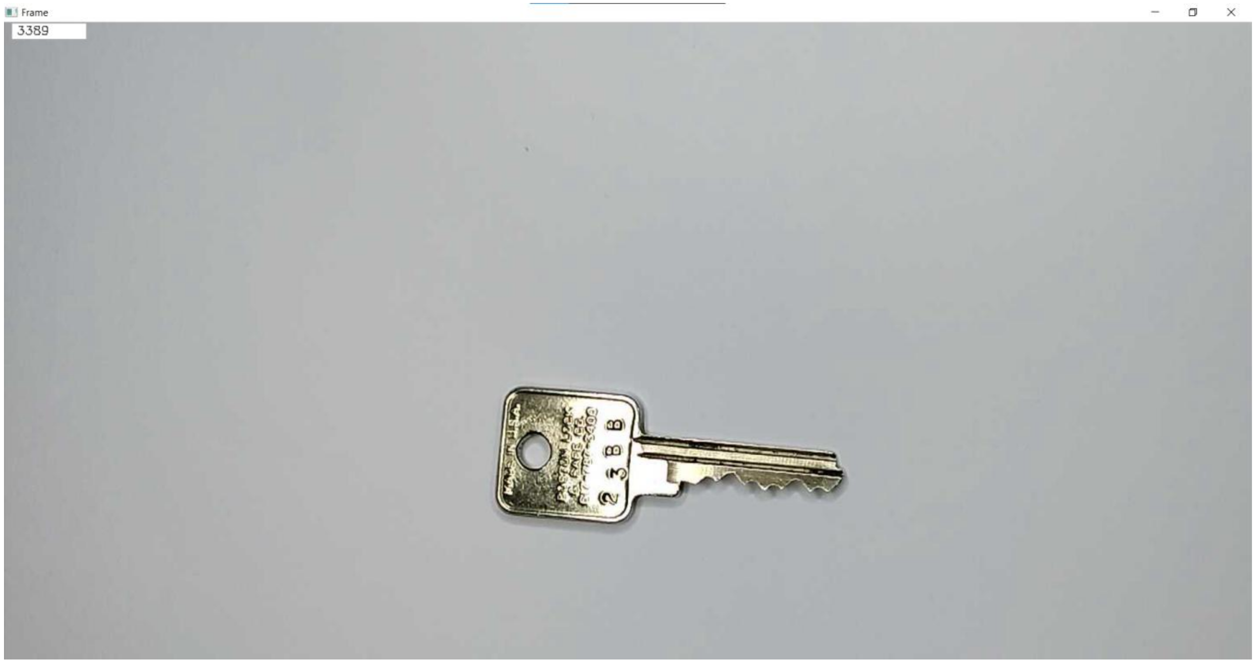
Also note, that the background subtraction implemented assumes the first frame to be the background to be subtracted from the images in the consequent frames that have the object. So it has to be ensured that the initial frame on running the program does not have the object.

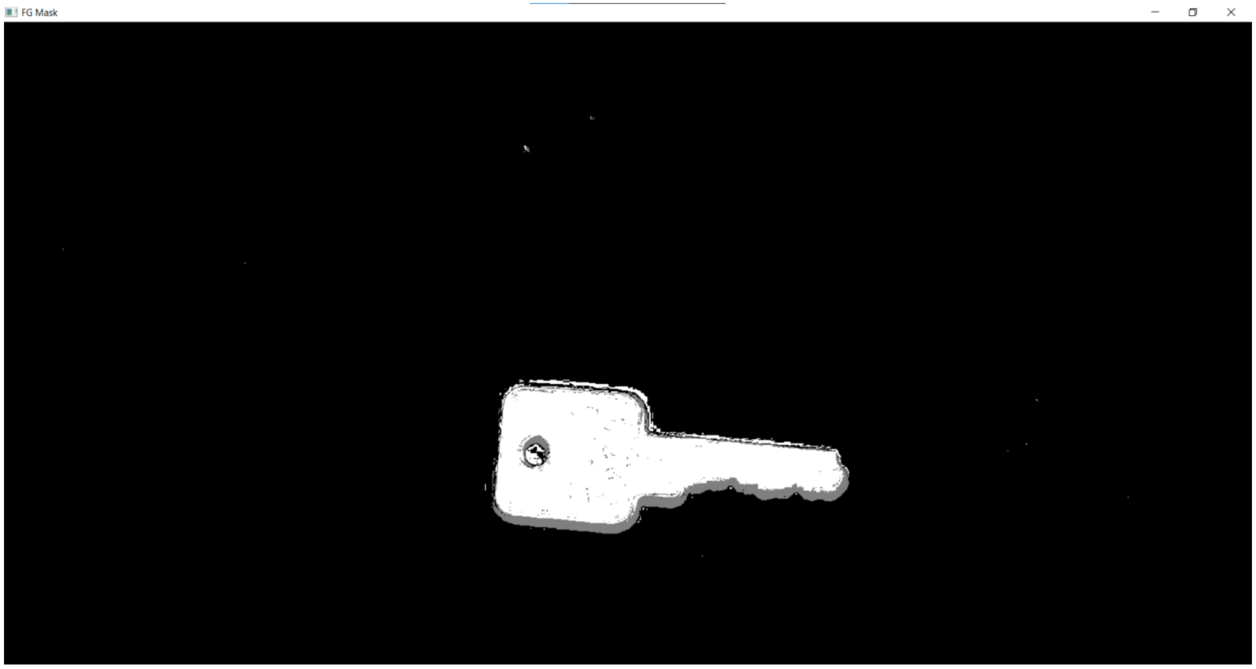
RESULTS: Task 1- Thresholding the input video

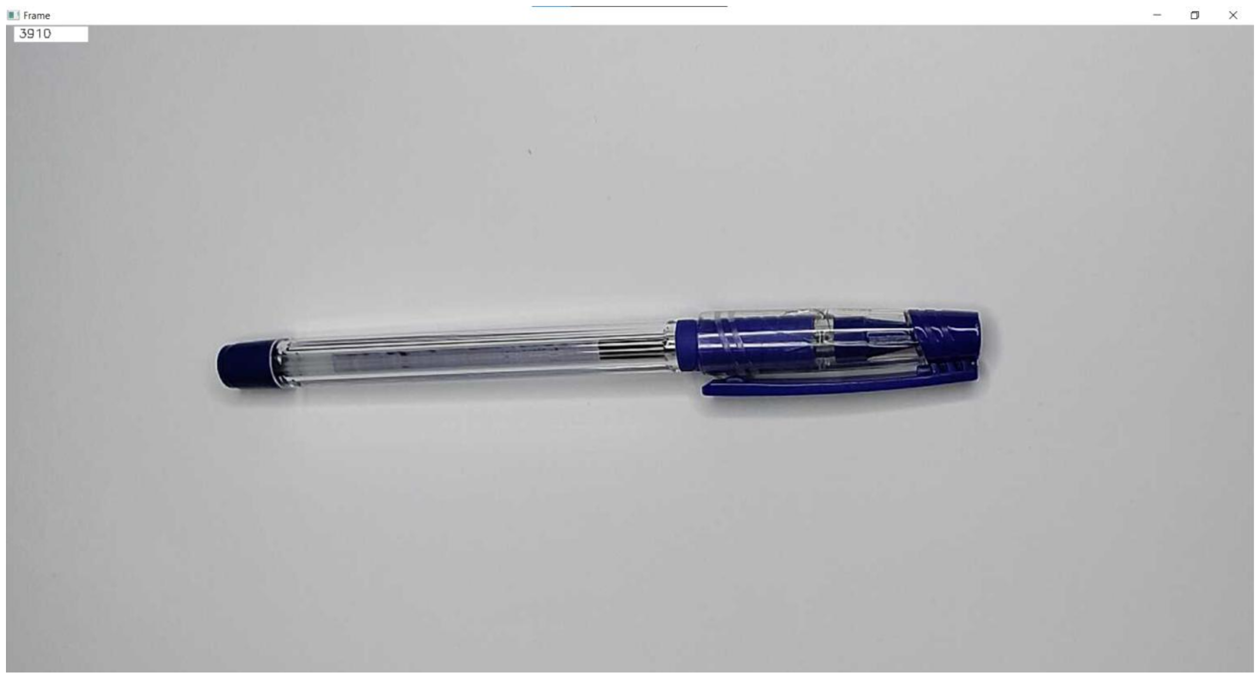
Required Image 1: 3 examples of the thresholded objects

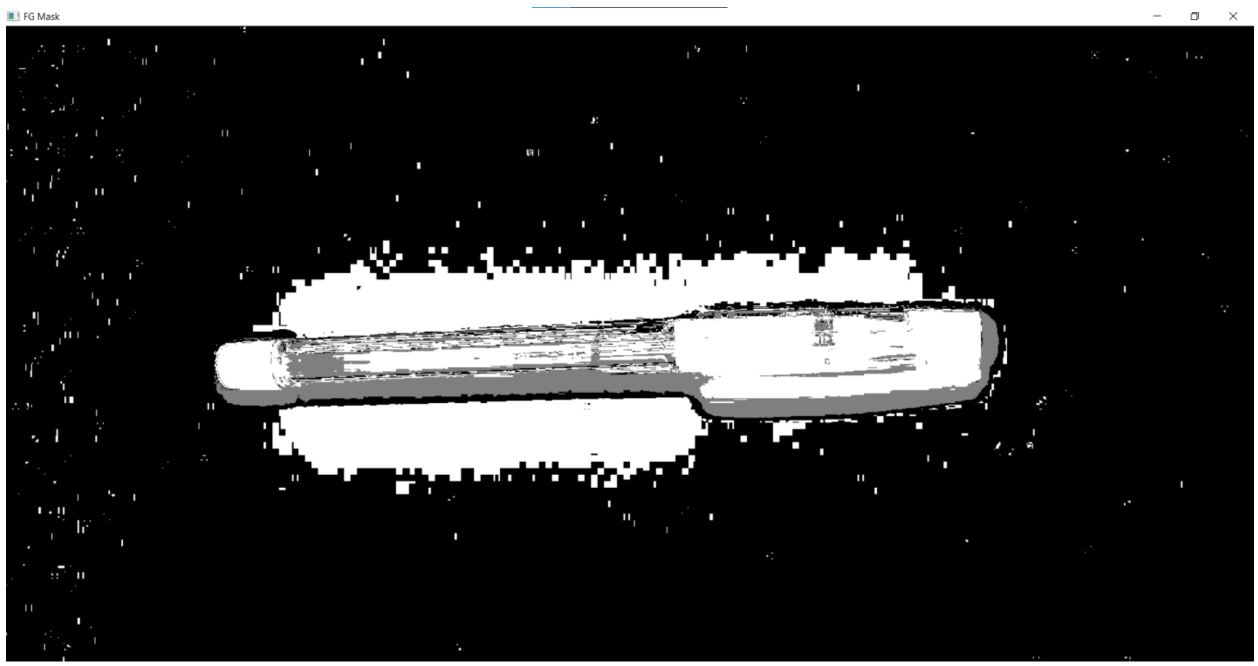

Frame
2901


FG Mask


Frame
3389

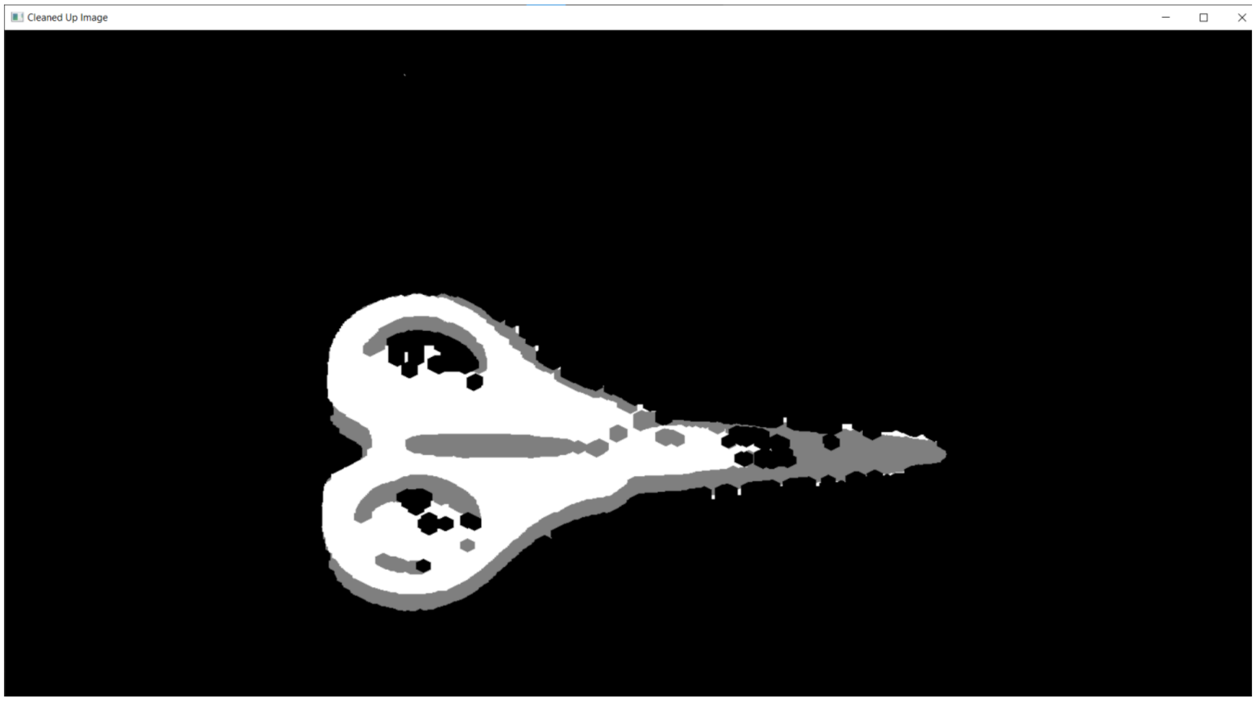

FG Mask

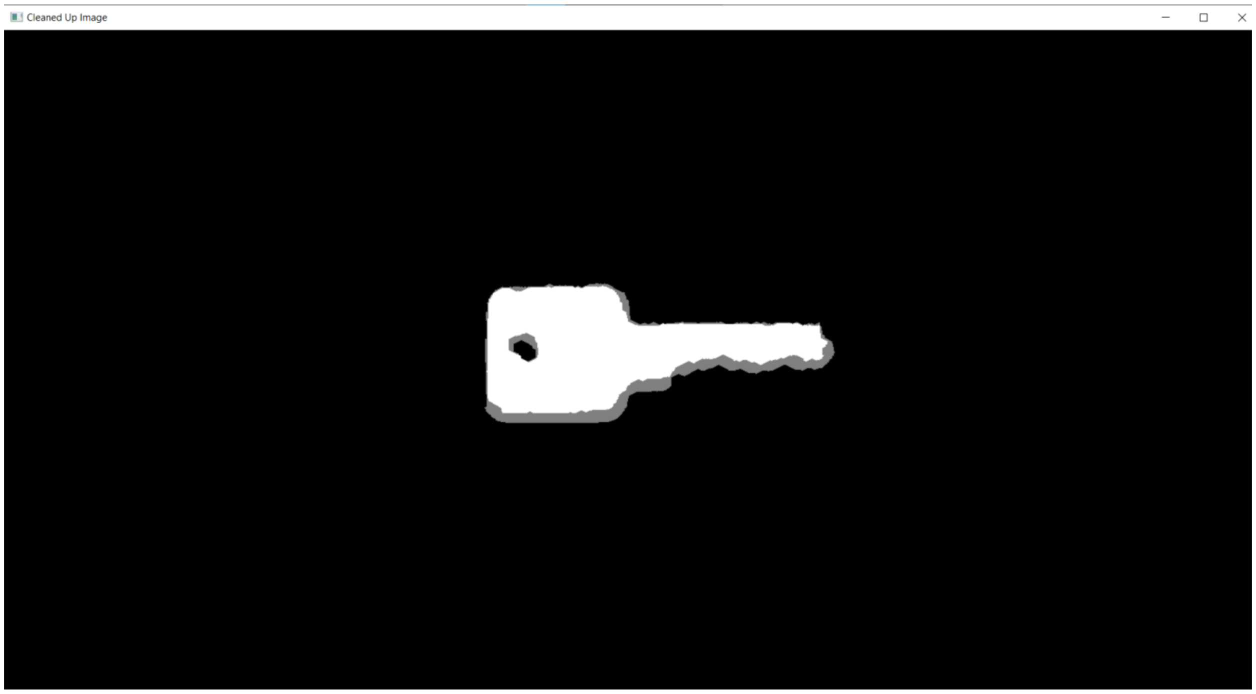


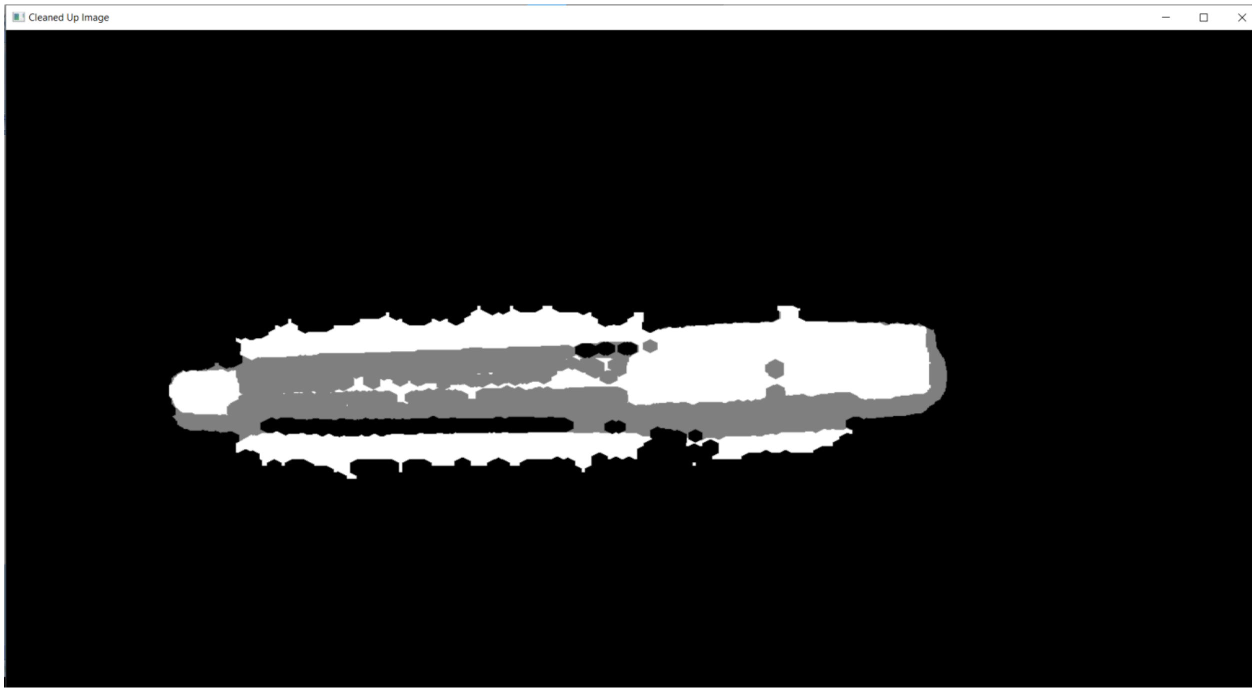


Task 2- Clean up the binary image

Required Image 2: 3 examples of the cleaned up binary image of objects

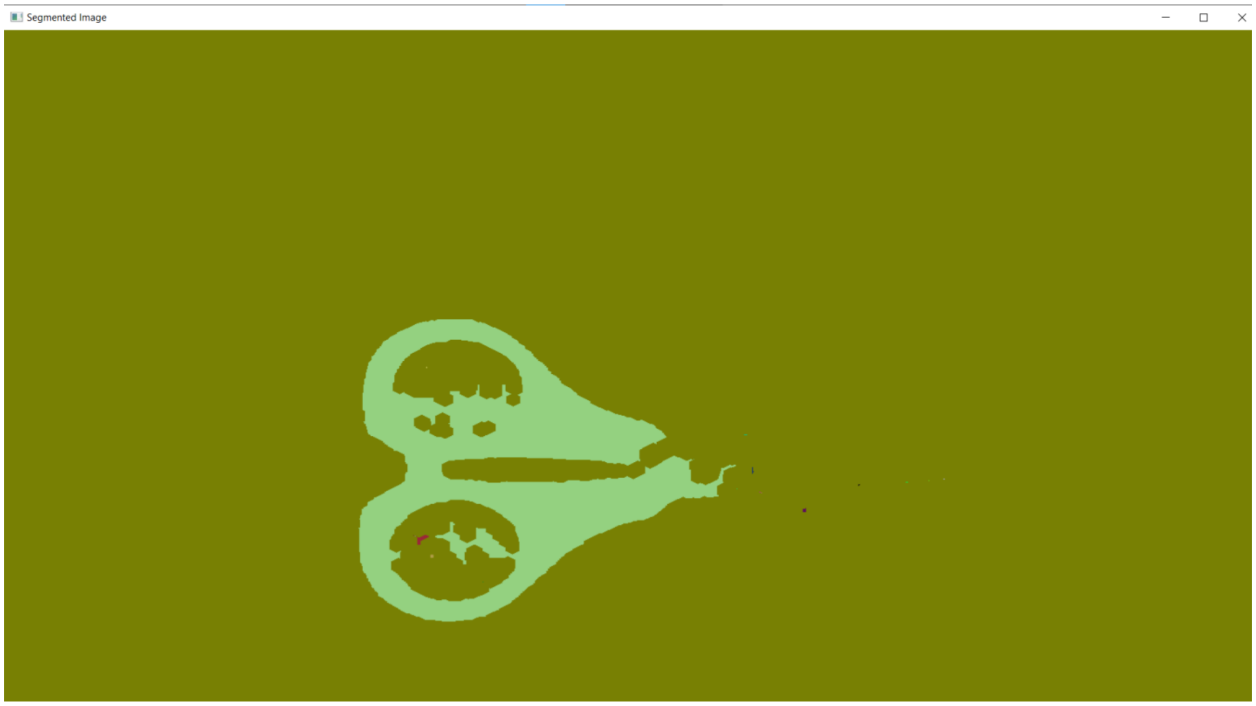

Cleaned Up Image

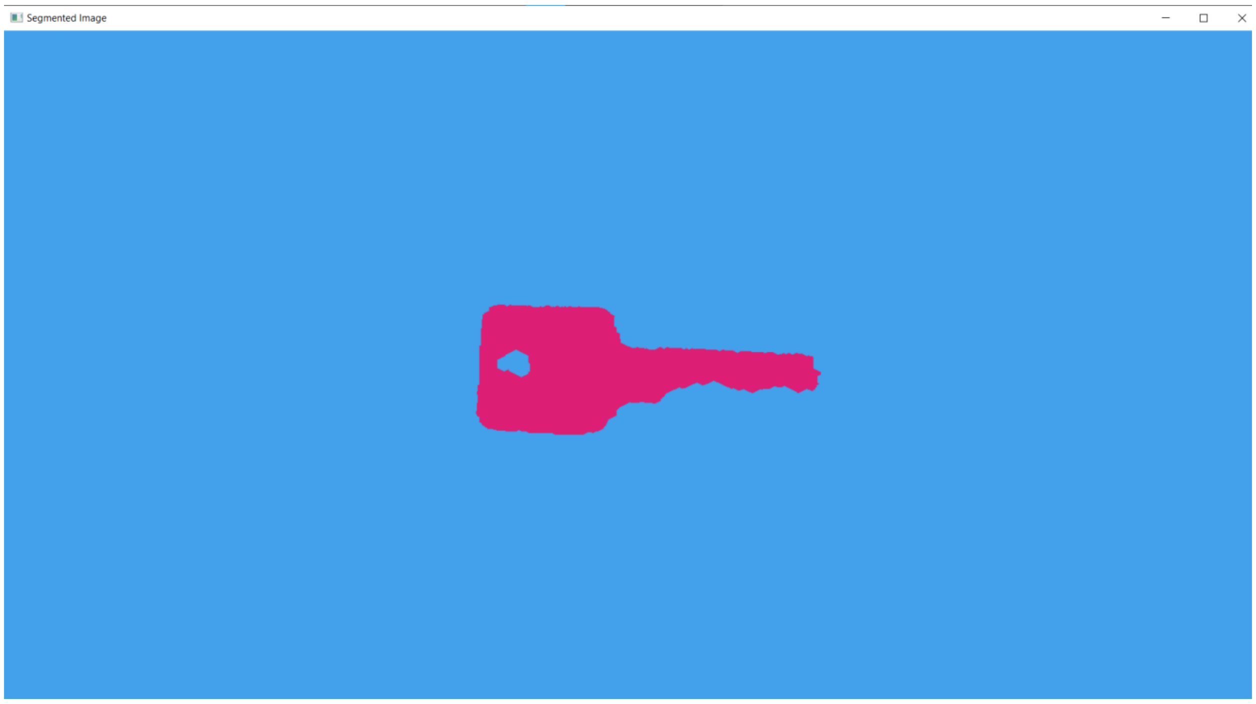

Cleaned Up Image

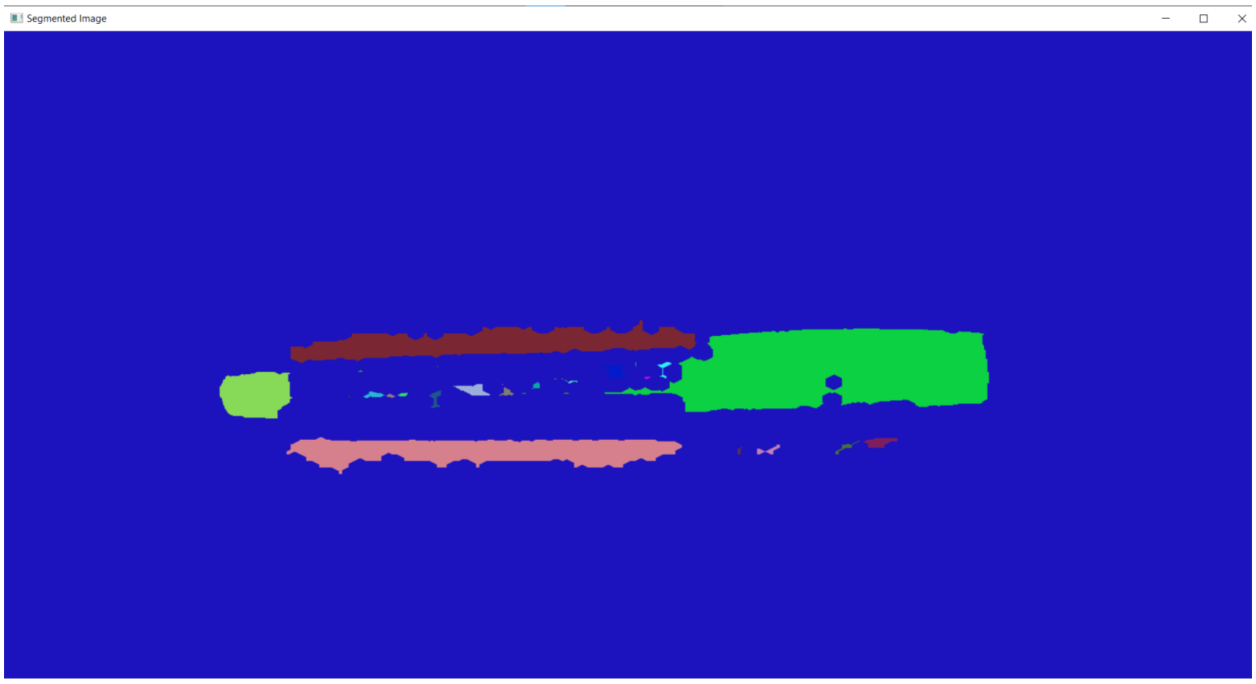


Task 3- Segment the image into regions

Required Image 3: 3 examples of the region maps with each region in a different color

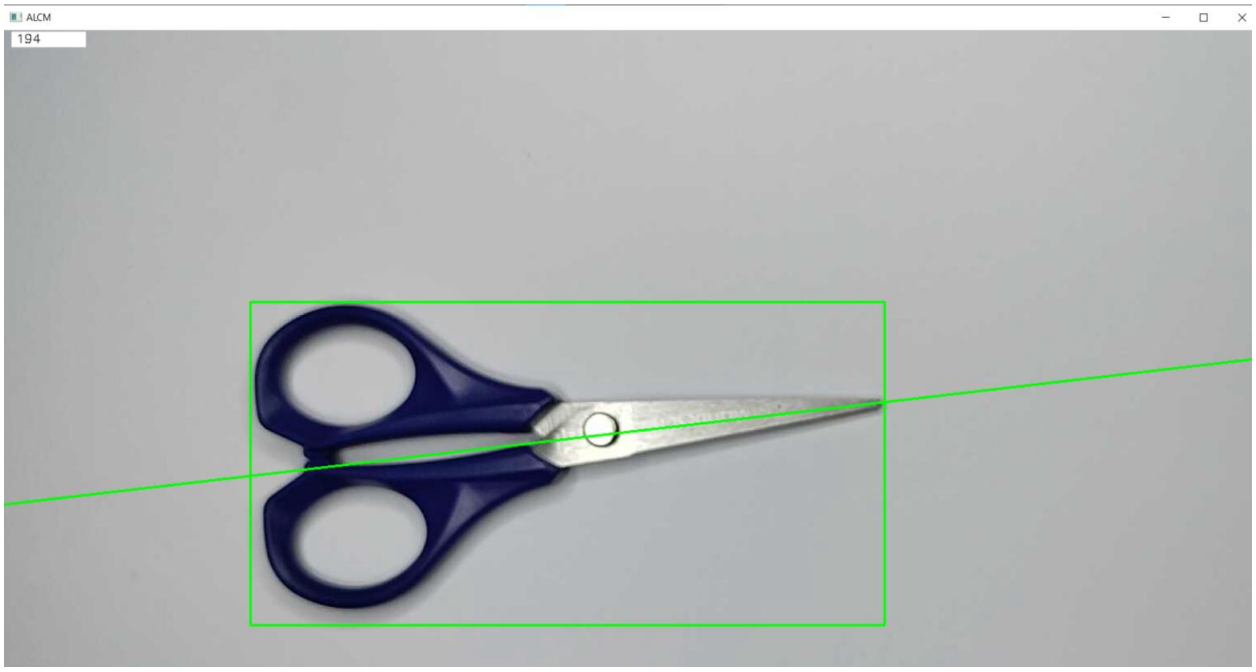


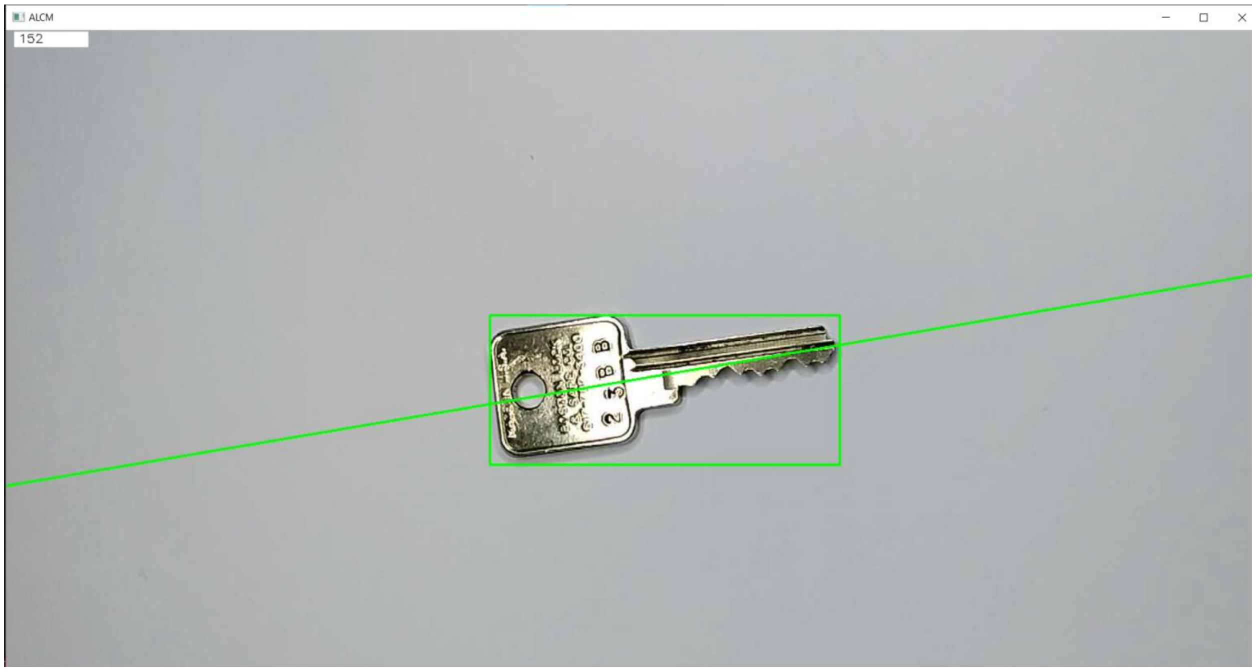


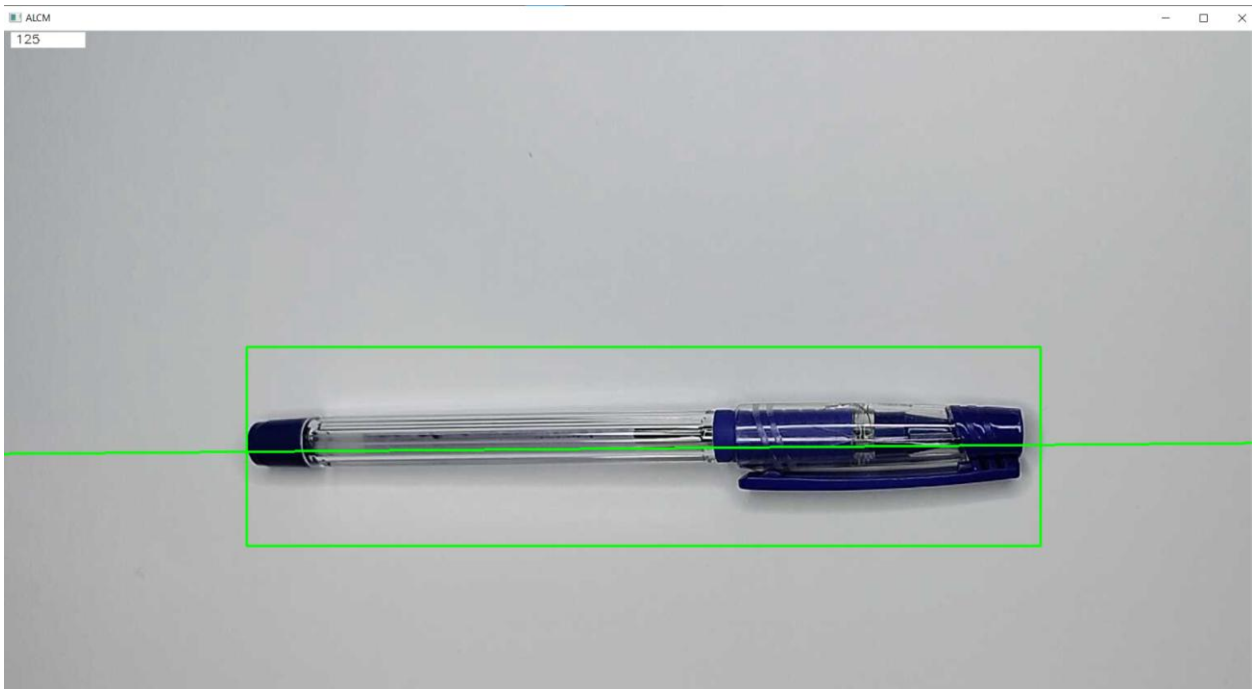


Task 4- Compute features for each major region

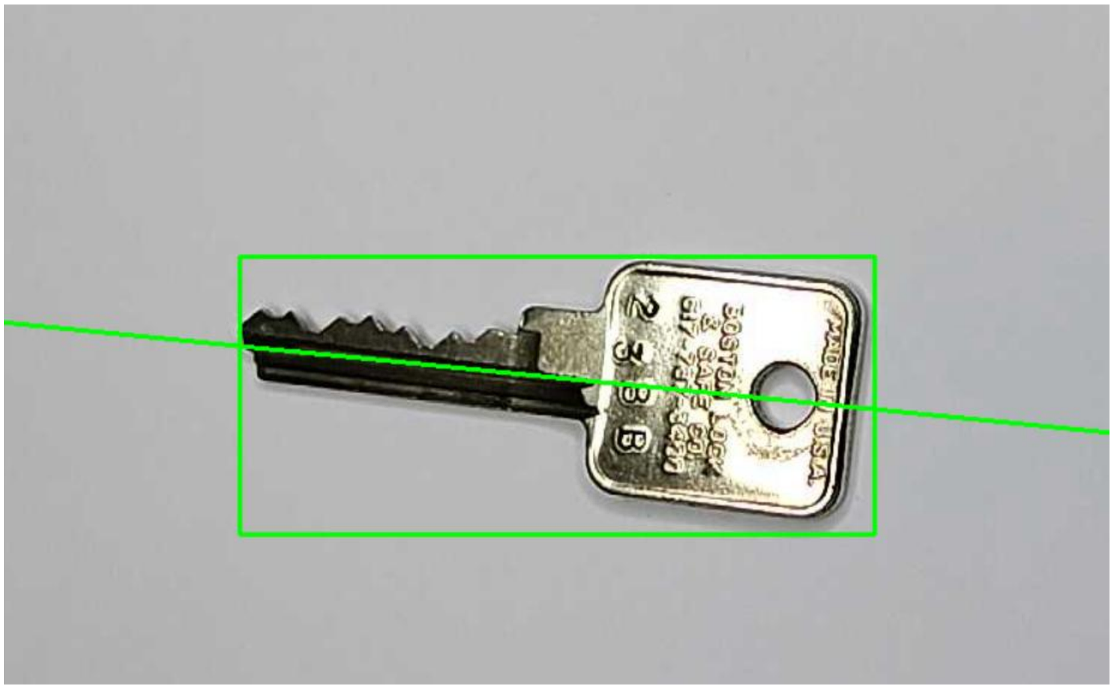
Required Image 4: 3 examples showing the axis of least central moment and the oriented bounding box

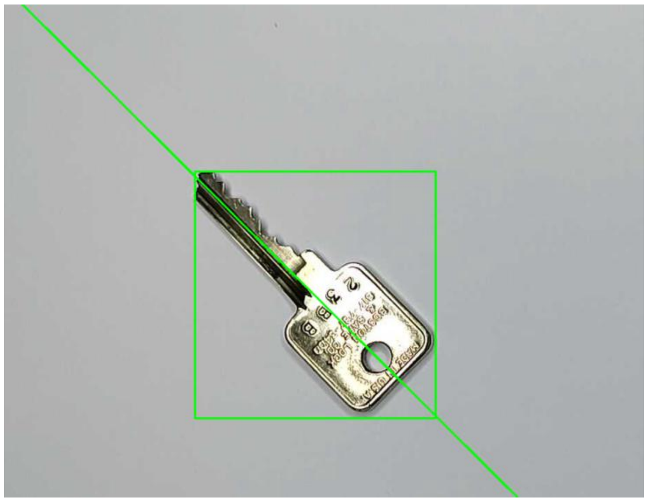

ALCM
194

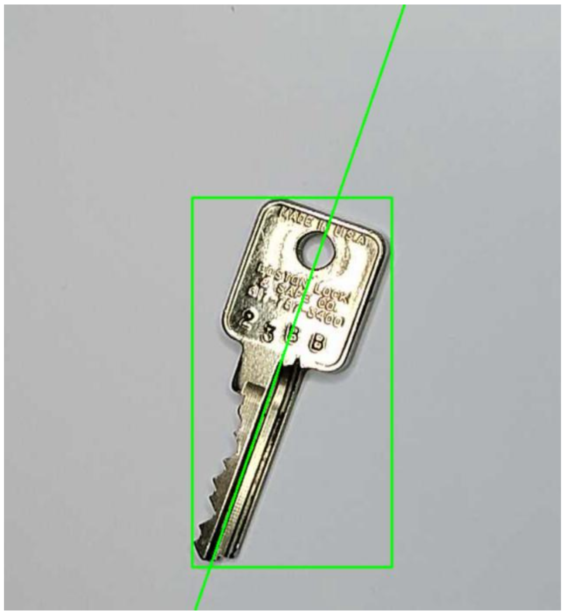

ALCM
152



We move the objects in the frame to ensure that the Hu moments and normalized central moment are printed out in output window with only little variant to have rotation, scale and translation invariant features. The axis of least central moment is displayed in real time on moving or rotating the object:

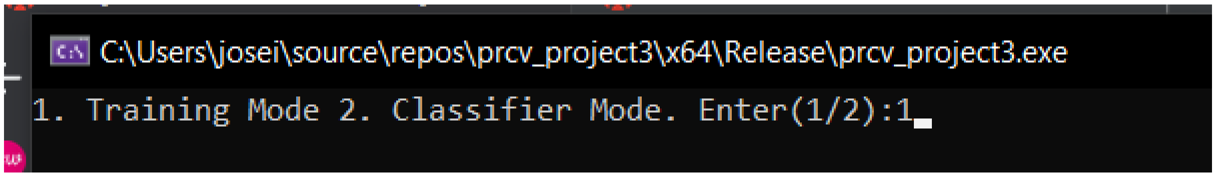






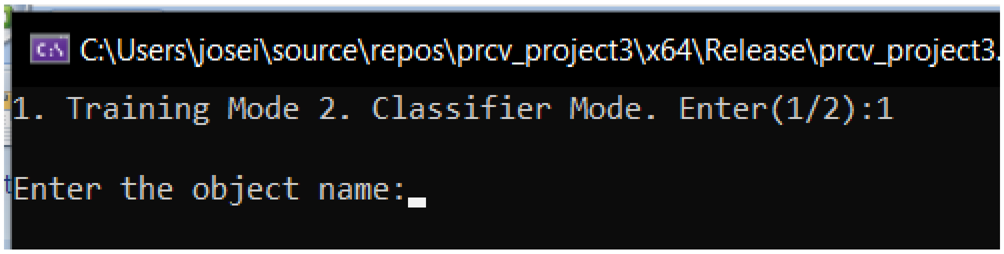
Task 5- Collect training data

System Working: The user is first given the following prompt:



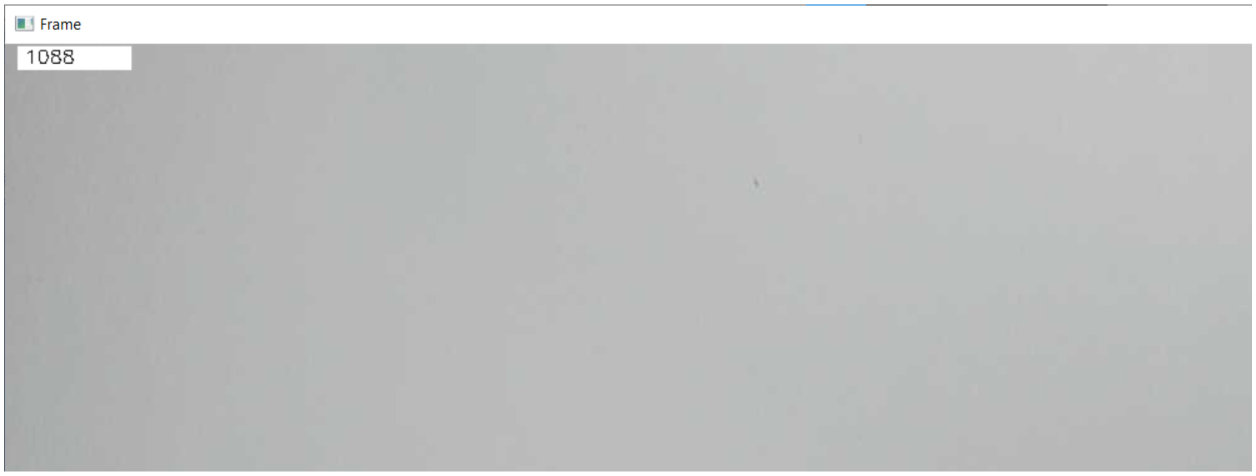
```
C:\Users\josei\source\repos\prcv_project3\x64\Release\prcv_project3.exe
1. Training Mode 2. Classifier Mode. Enter(1/2):1
```

On inputting 1, the user is prompted to enter the object label:



```
C:\Users\josei\source\repos\prcv_project3\x64\Release\prcv_project3.
1. Training Mode 2. Classifier Mode. Enter(1/2):1

Enter the object name:
```

On inputting the object name and hitting enter key, the original color image(with frame number) and the foreground mask image opens up in cv::imshow windows

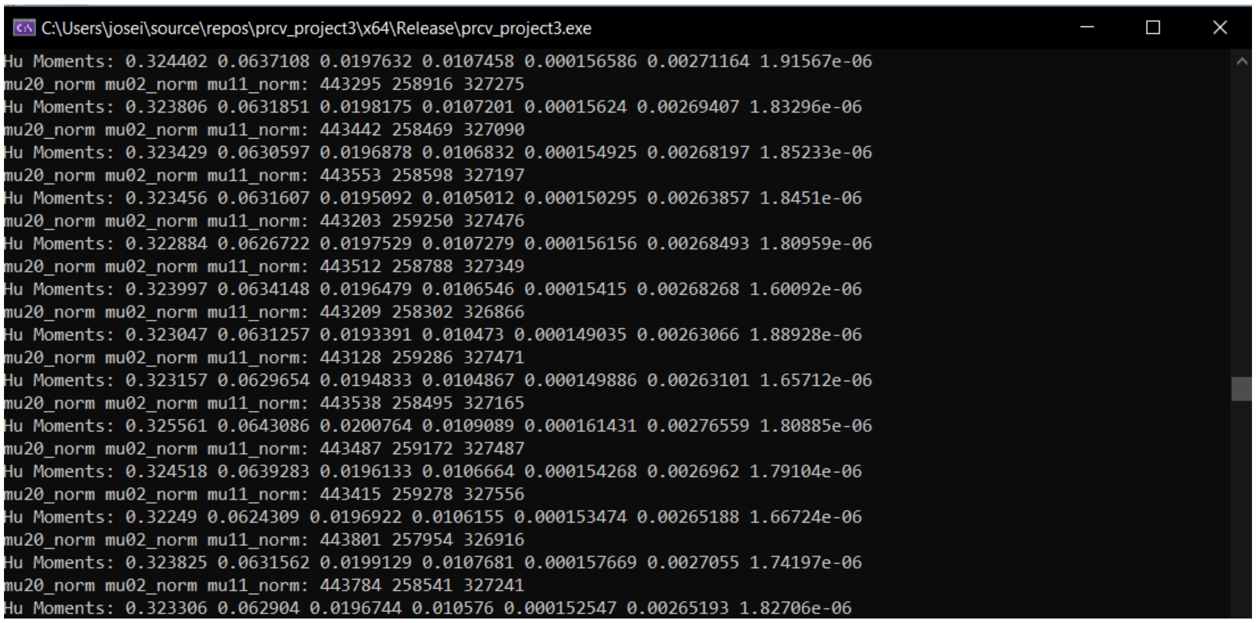


NOTE: the background subtraction implemented assumes the first frame to be the background to be subtracted from the images in the consequent frames that have the object. So it has to be ensured that the initial frame on running the program does not have the object.

Further, after the frame number is >1, the object is placed until the camera image is properly focused and then the user has to hit ‘q’ for the following cv::imshow windows to be displayed:

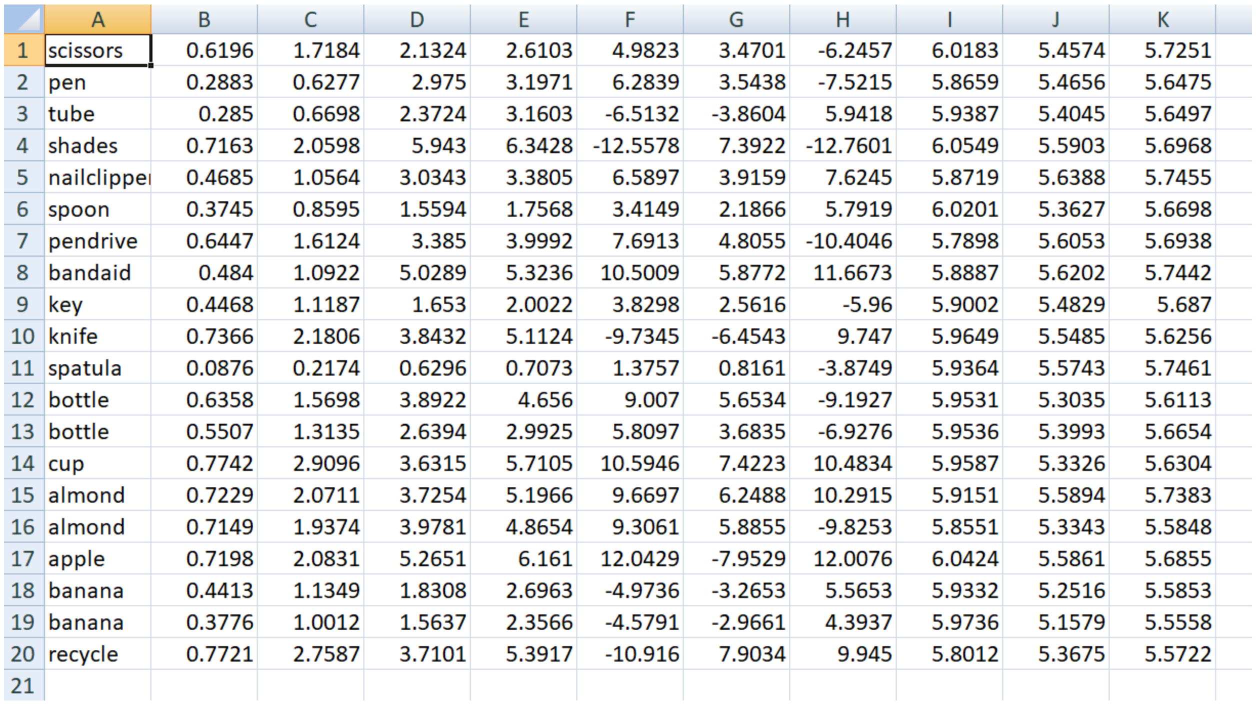
- Cleaned up image(after closing and opening morphological operations)
- bounding box with the axis(ALCM)

The Hu moments and normalized central moments calculated are printed in the terminal console:



```
C:\Users\josei\source\repos\prcv_project3\x64\Release\prcv_project3.exe
Hu Moments: 0.324402 0.0637108 0.0197632 0.0107458 0.000156586 0.00271164 1.91567e-06
mu20_norm mu02_norm mu11_norm: 443295 258916 327275
Hu Moments: 0.323806 0.0631851 0.0198175 0.0107201 0.00015624 0.00269407 1.83296e-06
mu20_norm mu02_norm mu11_norm: 443442 258469 327090
Hu Moments: 0.323429 0.0630597 0.0196878 0.0106832 0.000154925 0.00268197 1.85233e-06
mu20_norm mu02_norm mu11_norm: 443553 258598 327197
Hu Moments: 0.323456 0.0631607 0.0195092 0.0105012 0.000150295 0.00263857 1.8451e-06
mu20_norm mu02_norm mu11_norm: 443203 259250 327476
Hu Moments: 0.322884 0.0626722 0.0197529 0.0107279 0.000156156 0.00268493 1.80959e-06
mu20_norm mu02_norm mu11_norm: 443512 258788 327349
Hu Moments: 0.323997 0.0634148 0.0196479 0.0106546 0.00015415 0.00268268 1.60092e-06
mu20_norm mu02_norm mu11_norm: 443209 258302 326866
Hu Moments: 0.323047 0.0631257 0.0193391 0.010473 0.000149035 0.00263066 1.88928e-06
mu20_norm mu02_norm mu11_norm: 443128 259286 327471
Hu Moments: 0.323157 0.0629654 0.0194833 0.0104867 0.000149886 0.00263101 1.65712e-06
mu20_norm mu02_norm mu11_norm: 443538 258495 327165
Hu Moments: 0.325561 0.0643086 0.0200764 0.0109089 0.000161431 0.00276559 1.80885e-06
mu20_norm mu02_norm mu11_norm: 443487 259172 327487
Hu Moments: 0.324518 0.0639283 0.0196133 0.0106664 0.000154268 0.0026962 1.79104e-06
mu20_norm mu02_norm mu11_norm: 443415 259278 327556
Hu Moments: 0.32249 0.0624309 0.0196922 0.0106155 0.000153474 0.00265188 1.66724e-06
mu20_norm mu02_norm mu11_norm: 443801 257954 326916
Hu Moments: 0.323825 0.0631562 0.0199129 0.0107681 0.000157669 0.0027055 1.74197e-06
mu20_norm mu02_norm mu11_norm: 443784 258541 327241
Hu Moments: 0.323306 0.062904 0.0196744 0.010576 0.000152547 0.00265193 1.82706e-06
```

On hitting ‘q’ again, the most recent feature value of Hu moments followed by mu20_norm, mu02_norm and mu11_norm are appended to the csv file:



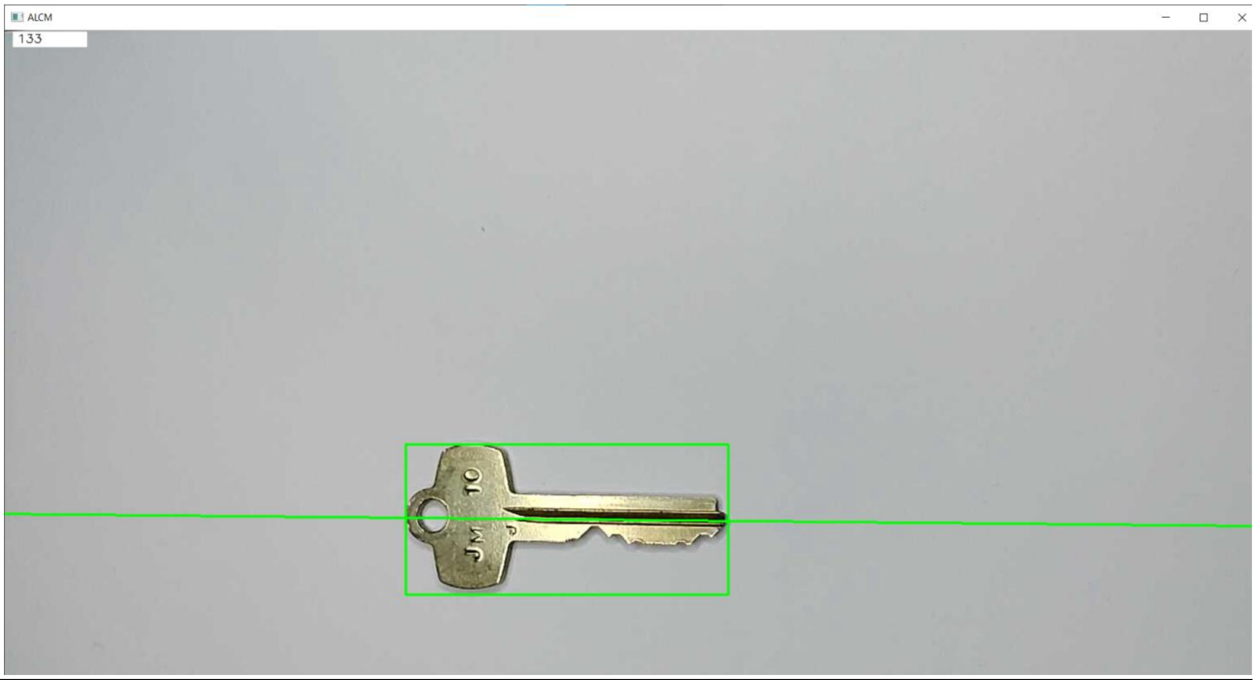
	A	B	C	D	E	F	G	H	I	J	K
1	scissors	0.6196	1.7184	2.1324	2.6103	4.9823	3.4701	-6.2457	6.0183	5.4574	5.7251
2	pen	0.2883	0.6277	2.975	3.1971	6.2839	3.5438	-7.5215	5.8659	5.4656	5.6475
3	tube	0.285	0.6698	2.3724	3.1603	-6.5132	-3.8604	5.9418	5.9387	5.4045	5.6497
4	shades	0.7163	2.0598	5.943	6.3428	-12.5578	7.3922	-12.7601	6.0549	5.5903	5.6968
5	nailclippei	0.4685	1.0564	3.0343	3.3805	6.5897	3.9159	7.6245	5.8719	5.6388	5.7455
6	spoon	0.3745	0.8595	1.5594	1.7568	3.4149	2.1866	5.7919	6.0201	5.3627	5.6698
7	pendrive	0.6447	1.6124	3.385	3.9992	7.6913	4.8055	-10.4046	5.7898	5.6053	5.6938
8	bandaid	0.484	1.0922	5.0289	5.3236	10.5009	5.8772	11.6673	5.8887	5.6202	5.7442
9	key	0.4468	1.1187	1.653	2.0022	3.8298	2.5616	-5.96	5.9002	5.4829	5.687
10	knife	0.7366	2.1806	3.8432	5.1124	-9.7345	-6.4543	9.747	5.9649	5.5485	5.6256
11	spatula	0.0876	0.2174	0.6296	0.7073	1.3757	0.8161	-3.8749	5.9364	5.5743	5.7461
12	bottle	0.6358	1.5698	3.8922	4.656	9.007	5.6534	-9.1927	5.9531	5.3035	5.6113
13	bottle	0.5507	1.3135	2.6394	2.9925	5.8097	3.6835	-6.9276	5.9536	5.3993	5.6654
14	cup	0.7742	2.9096	3.6315	5.7105	10.5946	7.4223	10.4834	5.9587	5.3326	5.6304
15	almond	0.7229	2.0711	3.7254	5.1966	9.6697	6.2488	10.2915	5.9151	5.5894	5.7383
16	almond	0.7149	1.9374	3.9781	4.8654	9.3061	5.8855	-9.8253	5.8551	5.3343	5.5848
17	apple	0.7198	2.0831	5.2651	6.161	12.0429	-7.9529	12.0076	6.0424	5.5861	5.6855
18	banana	0.4413	1.1349	1.8308	2.6963	-4.9736	-3.2653	5.5653	5.9332	5.2516	5.5853
19	banana	0.3776	1.0012	1.5637	2.3566	-4.5791	-2.9661	4.3937	5.9736	5.1579	5.5558
20	recycle	0.7721	2.7587	3.7101	5.3917	-10.916	7.9034	9.945	5.8012	5.3675	5.5722
21											

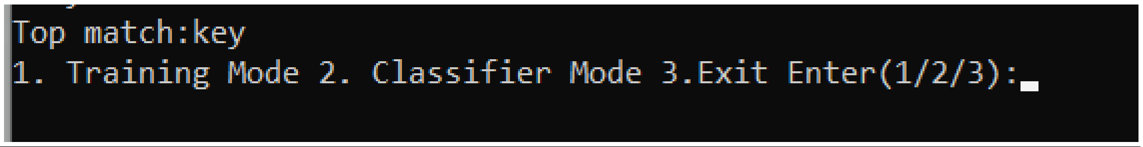
Hence, we train the system with objects to get the csv file as shown above

After this phase, the program loops back to the start of asking the user’s input to enter training (adding more objects) mode or classification mode.

Task 6- Classify new images

On entering the classification mode, the user has to wait till, after the first frame is formed, to place the object in foreground and get it focused in the camera. Then on hitting ‘q’, the bounding box is drawn along with moment calculation and scaled Euclidean distance between all existing objects in csv database file. Then the top match is printed in the terminal output:

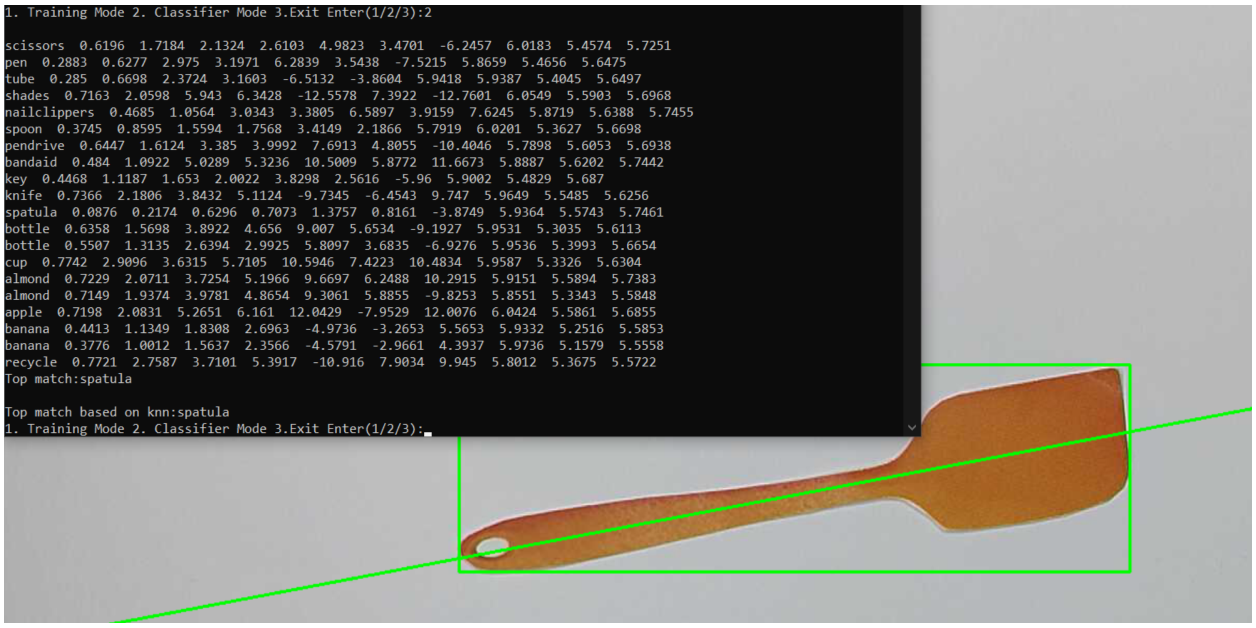




```
Top match:key
1. Training Mode 2. Classifier Mode 3.Exit Enter(1/2/3):
```

Task 7- Implement a different classifier

The KNN implementation in the code is performed during the classification phase prompted by the user itself



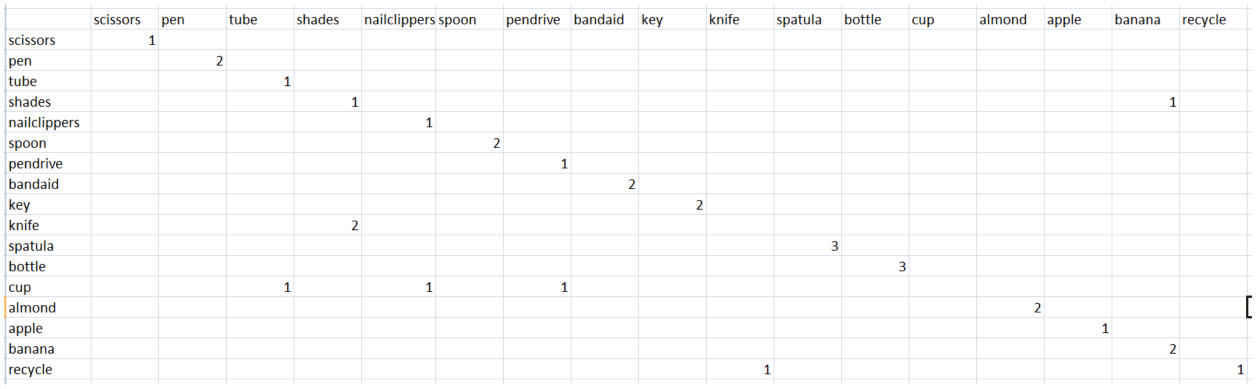
```
1. Training Mode 2. Classifier Mode 3.Exit Enter(1/2/3):2

scissors  0.6196  1.7184  2.1324  2.6103  4.9823  3.4701  -6.2457  6.0183  5.4574  5.7251
pen  0.2883  0.6277  2.975  3.1971  6.2839  3.5438  -7.5215  5.8659  5.4656  5.6475
tube  0.285  0.6698  2.3724  3.1603  -6.5132  -3.8604  5.9418  5.9387  5.4045  5.6497
shades  0.7163  2.0598  5.943  6.3428  -12.5578  7.3922  -12.7601  6.0549  5.5903  5.6968
nailclippers  0.4685  1.0564  3.0343  3.3805  6.5897  3.9159  7.6245  5.8719  5.6388  5.7455
spoon  0.3745  0.8595  1.5594  1.7568  3.4149  2.1866  5.7919  6.0201  5.3627  5.6698
pendrive  0.6447  1.6124  3.385  3.9992  7.6913  4.8055  -10.4046  5.7898  5.6053  5.6938
bandaid  0.484  1.0922  5.0289  5.3236  10.5009  5.8772  11.6673  5.8887  5.6202  5.7442
key  0.4468  1.1187  1.653  2.0022  3.8298  2.5616  -5.96  5.9002  5.4829  5.687
knife  0.7366  2.1806  3.8432  5.1124  -9.7345  -6.4543  9.747  5.9649  5.5485  5.6256
spatula  0.0876  0.2174  0.6296  0.7073  1.3757  0.8161  -3.8749  5.9364  5.5743  5.7461
bottle  0.6358  1.5698  3.8922  4.656  9.007  5.6534  -9.1927  5.9531  5.3035  5.6113
bottle  0.5507  1.3135  2.6394  2.9925  5.8097  3.6835  -6.9276  5.9536  5.3993  5.6654
cup  0.7742  2.9096  3.6315  5.7105  10.5946  7.4223  10.4834  5.9587  5.3326  5.6304
almond  0.7229  2.0711  3.7254  5.1966  9.6697  6.2488  10.2915  5.9151  5.5894  5.7383
almond  0.7149  1.9374  3.9781  4.8654  9.3061  5.8855  -9.8253  5.8551  5.3343  5.5848
apple  0.7198  2.0831  5.2651  6.161  12.0429  -7.9529  12.0076  6.0424  5.5861  5.6855
banana  0.4413  1.1349  1.8308  2.6963  -4.9736  -3.2653  5.5653  5.9332  5.2516  5.5853
banana  0.3776  1.0012  1.5637  2.3566  -4.5791  -2.9661  4.3937  5.9736  5.1579  5.5558
recycle  0.7721  2.7587  3.7101  5.3917  -10.916  7.9034  9.945  5.8012  5.3675  5.5722
Top match:spatula

Top match based on knn:spatula
1. Training Mode 2. Classifier Mode 3.Exit Enter(1/2/3):
```

Task 8- Evaluation of performance of system

The following is the confusion matrix where the rows of object name are actual inputs to system and the column inputs give the corresponding number of time different unique objects(of the same kind) were labeled by the system

	scissors	pen	tube	shades	nailclippers	spoon	pendrive	bandaid	key	knife	spatula	bottle	cup	almond	apple	banana	recycle
scissors	1																
pen		2															
tube					1												
shades				1												1	
nailclippers					1												
spoon						2											
pendrive							1										
bandaid								2									
key									2								
knife				2													
spatula											3						
bottle										1		2					
cup			1		1		1										
almond														2			
apple															1		
banana																2	
recycle										1							1

Confusion matrix of the KNN classifier output is as shown:

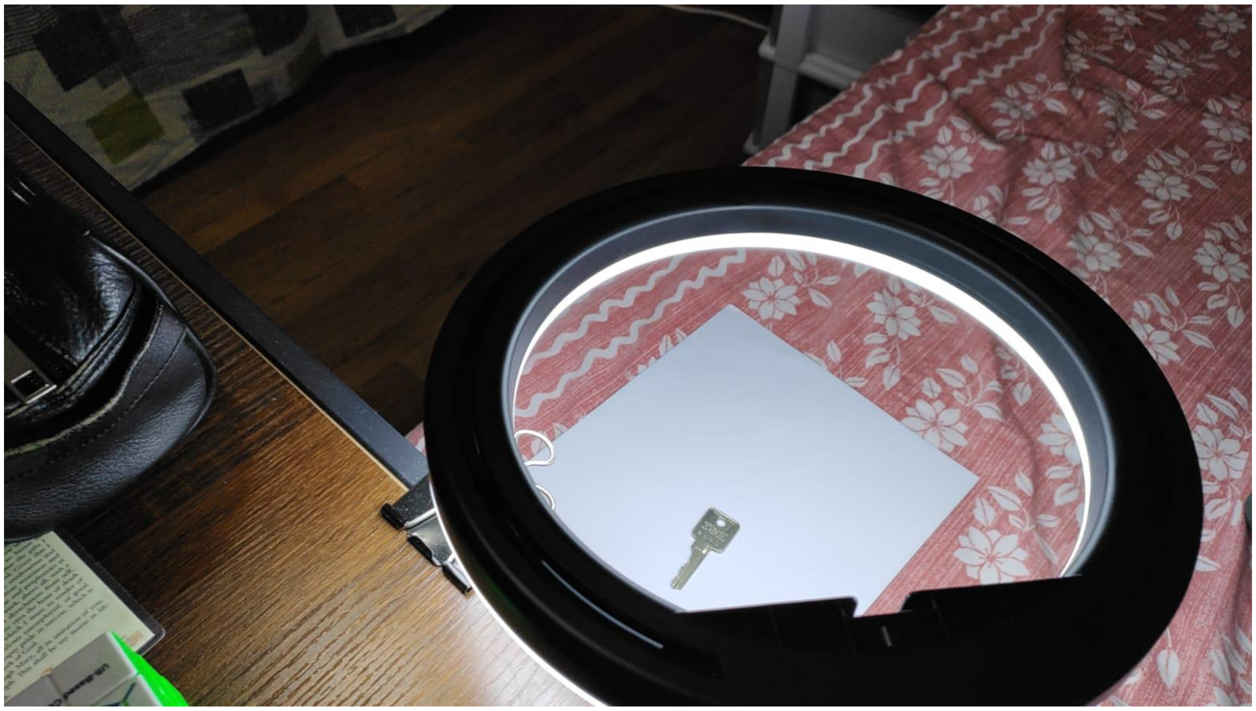


	scissors	pen	tube	shades	nailclippers	spoon	pendrive	bandaid	key	knife	spatula	bottle	cup	almond	apple	banana	recycle
scissors	1																
pen		2															
tube			1														
shades				1												1	
nailclippers					1												
spoon						2											
pendrive							1										
bandaid								2									
key									2								
knife				2													
spatula											3						
bottle												3					
cup			1		1		1										
almond														2			
apple															1		
banana																2	
recycle										1							1

As we see the implemented system is not perfect. There are quite a few objects that are being mislabeled. The implemented system also does not take into account for objects it has not be trained for. Even the KNN classifier does not seem to label all the shown objects correctly. But it is a step in the right direction of accuracy and scaled distance for feature matching.

Task 9 – Video of system running and classifying objects

System setup:



Video links to demos:

Training demo:

https://drive.google.com/file/d/1_DIEgdcOI3NPqtHMxkmJfvx47U4PrIli/view?usp
=sharing

Classification demo:
https://drive.google.com/file/d/18LN3rVUEHMGbmukVcy0qCd2wCsy73jBz/view?
usp=sharing

REFLECTION:

The project successfully implemented a real-time object recognition system using the OpenCV package. The system utilized a phone camera to capture image frames and compute rotation, scale, and translation invariant features of the object. The system also implemented a 2-pass algorithm for segmentation and used OpenCV library functions for other tasks. The user was prompted to enter inputs to go into a Training or Classification mode, where they could either compute features for new object labels or classify objects in real-time. The system utilized K-NN implementation of distance matching and a manually calculated confusion matrix to evaluate performance. Additionally, paper cut-outs of objects were added to the database to test the system further. Overall, the project successfully implemented a functional object recognition system with the ability to add new object labels and test against a variety of objects.

ACKNOWLEDGEMENT AND REFERENCES:

- Computer Vision: Algorithms and Applications, 2nd ed. Links to an external site., Rick Szeliski
- Shape matching using Hu moments
- Fourier descriptors – OpenCV (Extra reading)