CSE463 Introduction to Computer Vision HW2 Report

Abdullah Küsgülü
1801042606

Selected Objects

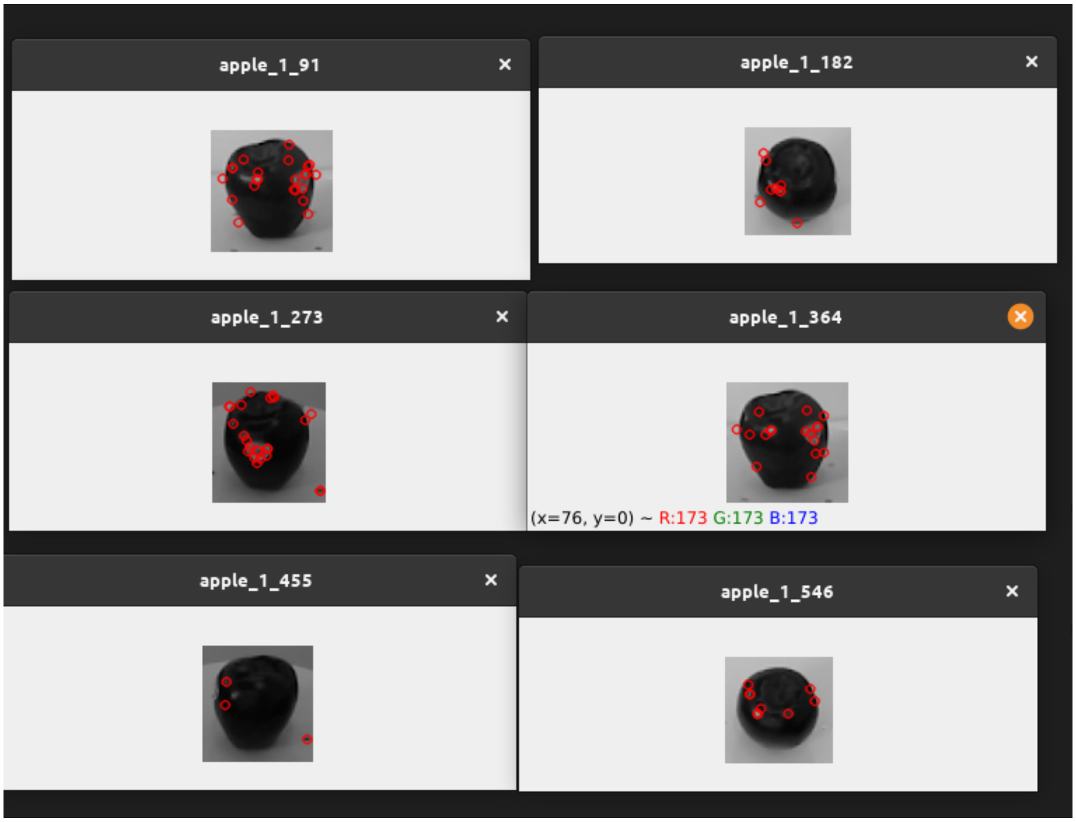
I have selected the following 10 object from the database specified in the PDF document:

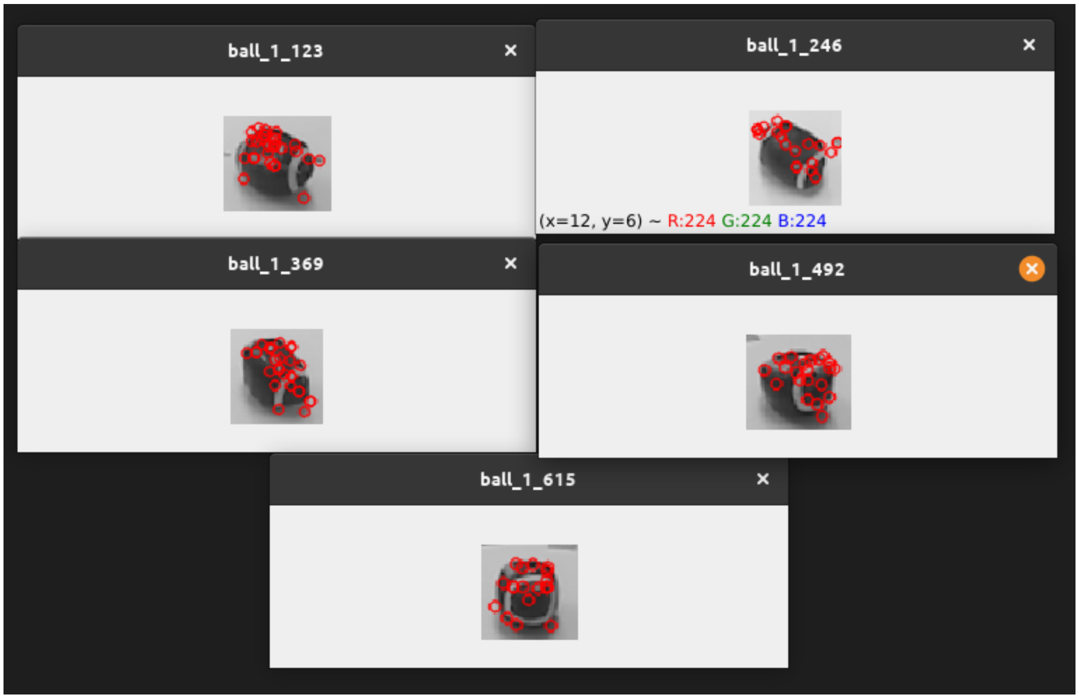
apple_1, ball_1, banana_1, bell_pepper_1, binder_1, bowl_1, calculator_1, camera_1, cap_1, cell_phone_1.

I have tidied this chosen subset by removing the depth related files. You can find the subset I used in this Google Drive link.

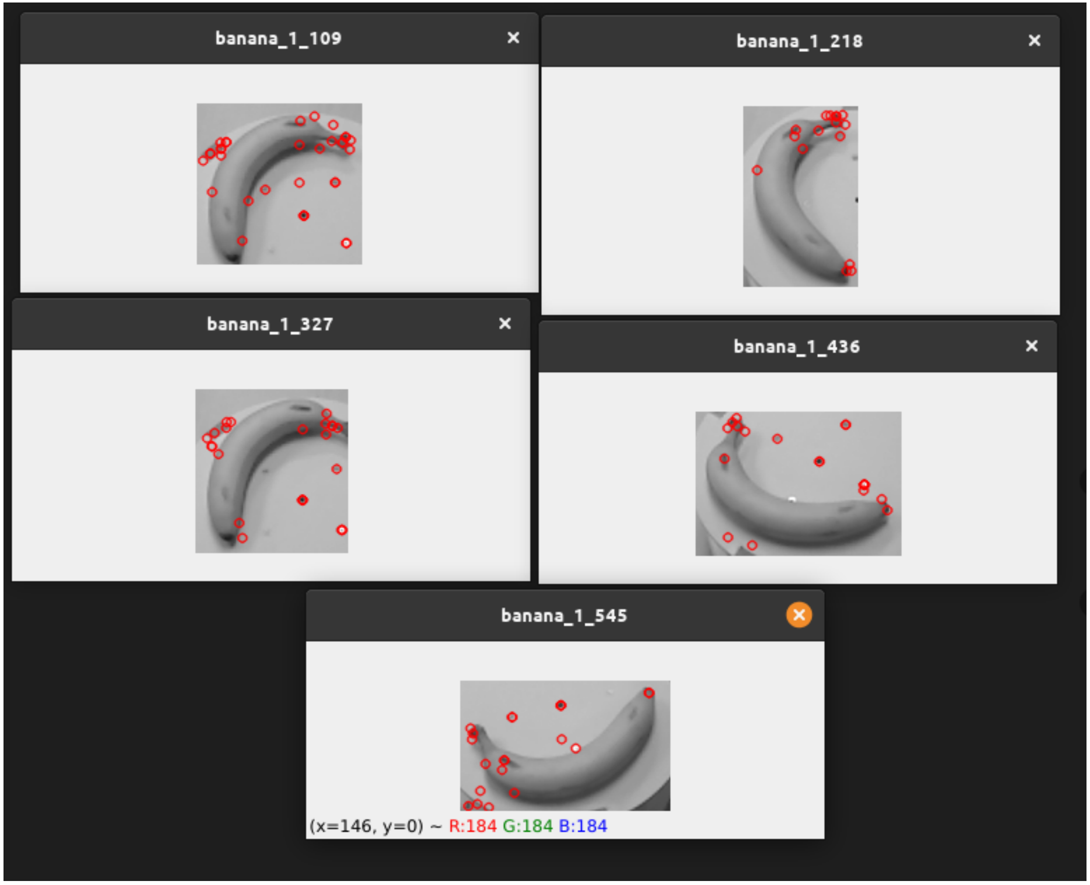
Features Detected on the Objects

I am putting only the keypoints found by SIFT to make the report not too long:

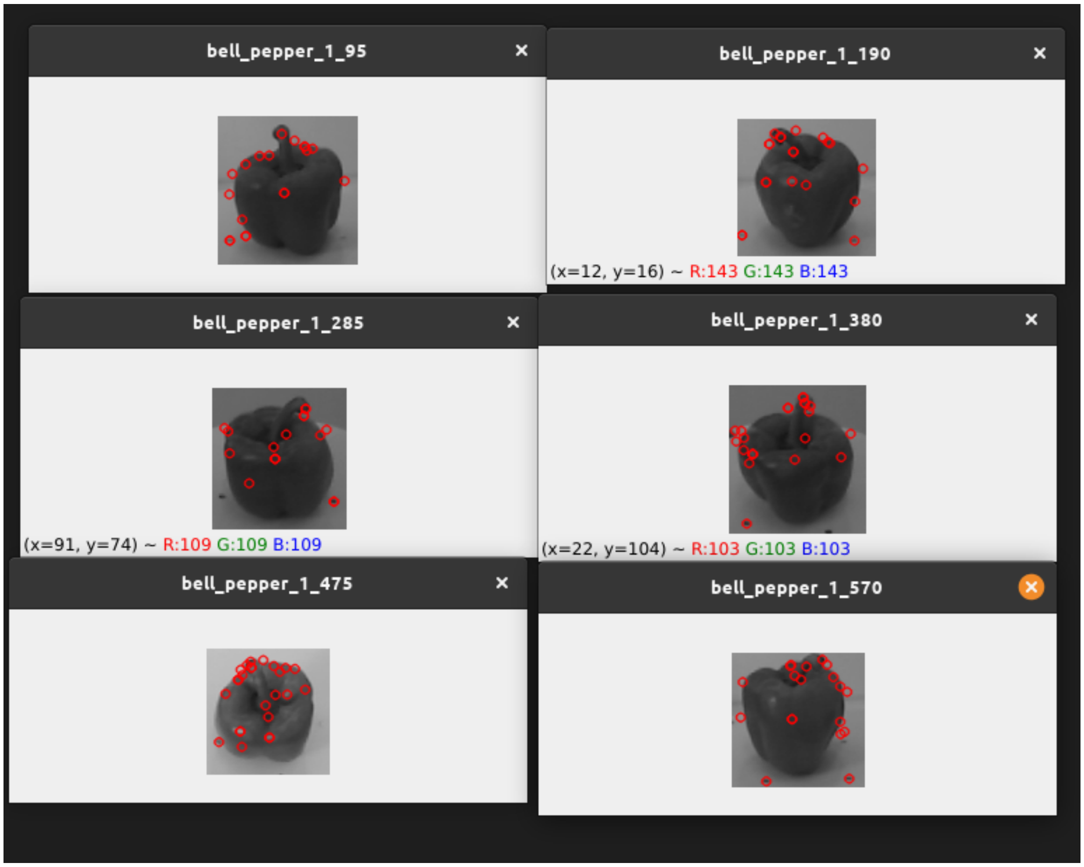




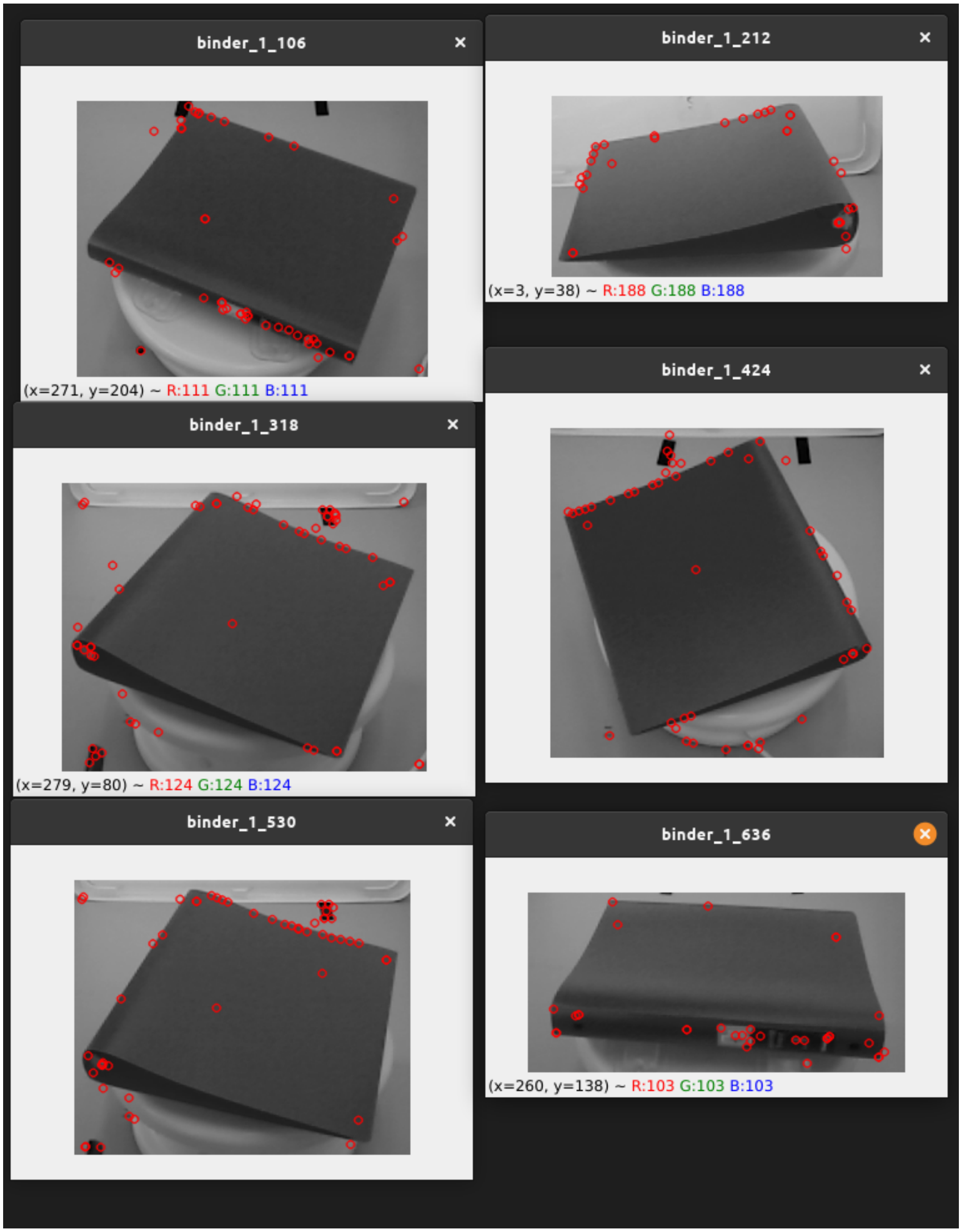
(x=12, y=6) ~ R:224 G:224 B:224



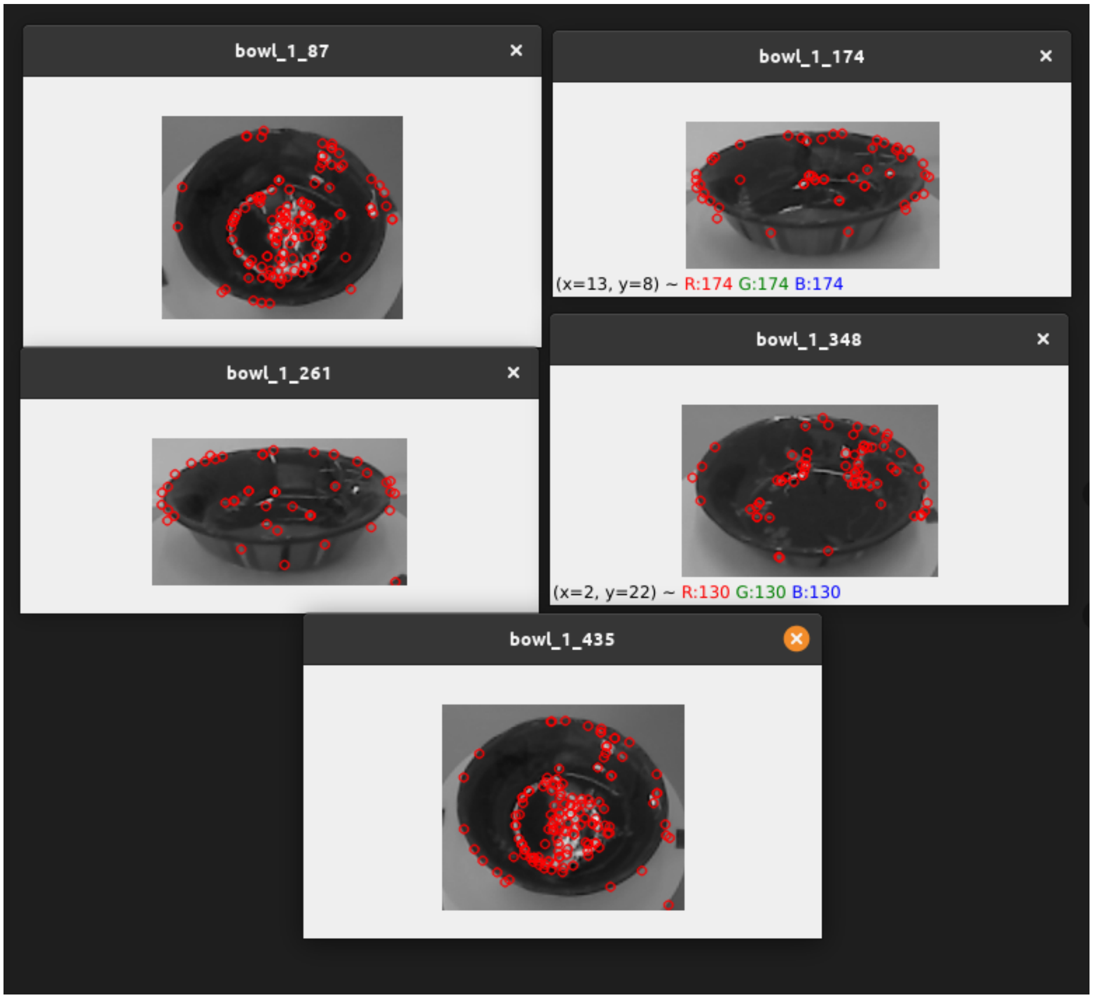
(x=146, y=0) ~ R:184 G:184 B:184



```
(x=12, y=16) ~ R:143 G:143 B:143
(x=91, y=74) ~ R:109 G:109 B:109
(x=22, y=104) ~ R:103 G:103 B:103
```

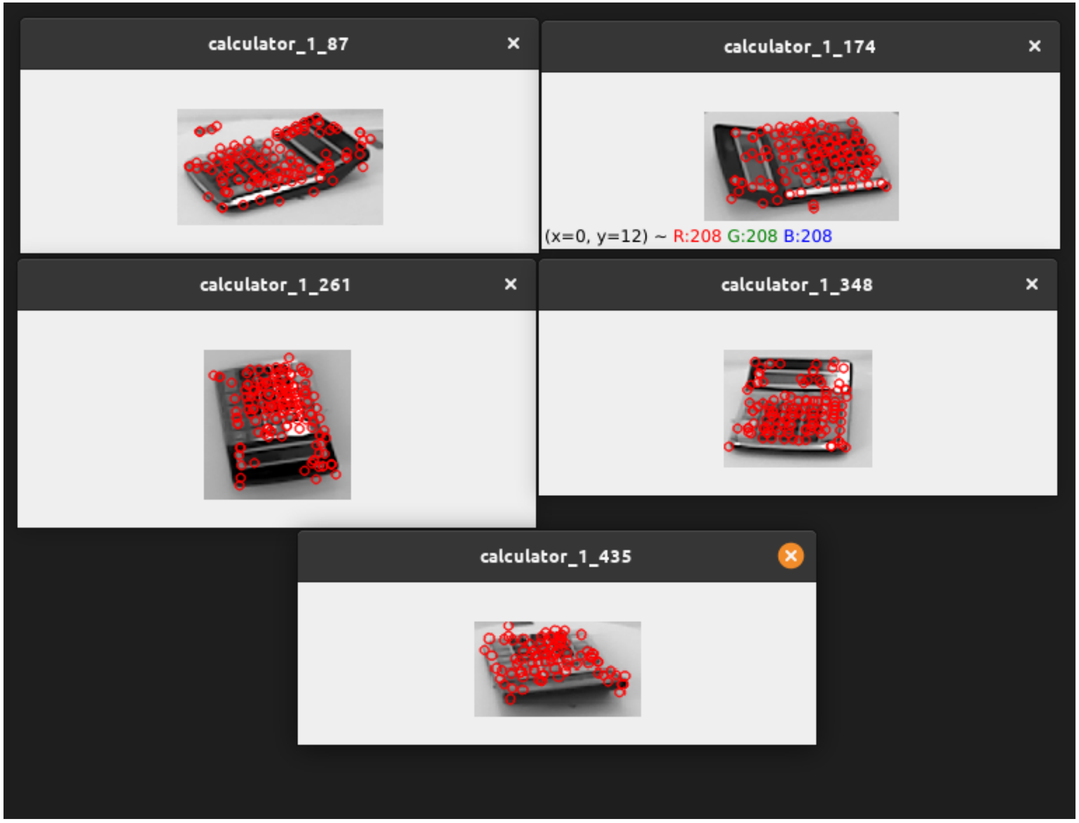


```
(x=271, y=204) ~ R:111 G:111 B:111
(x=3, y=38) ~ R:188 G:188 B:188
(x=279, y=80) ~ R:124 G:124 B:124
(x=260, y=138) ~ R:103 G:103 B:103
```

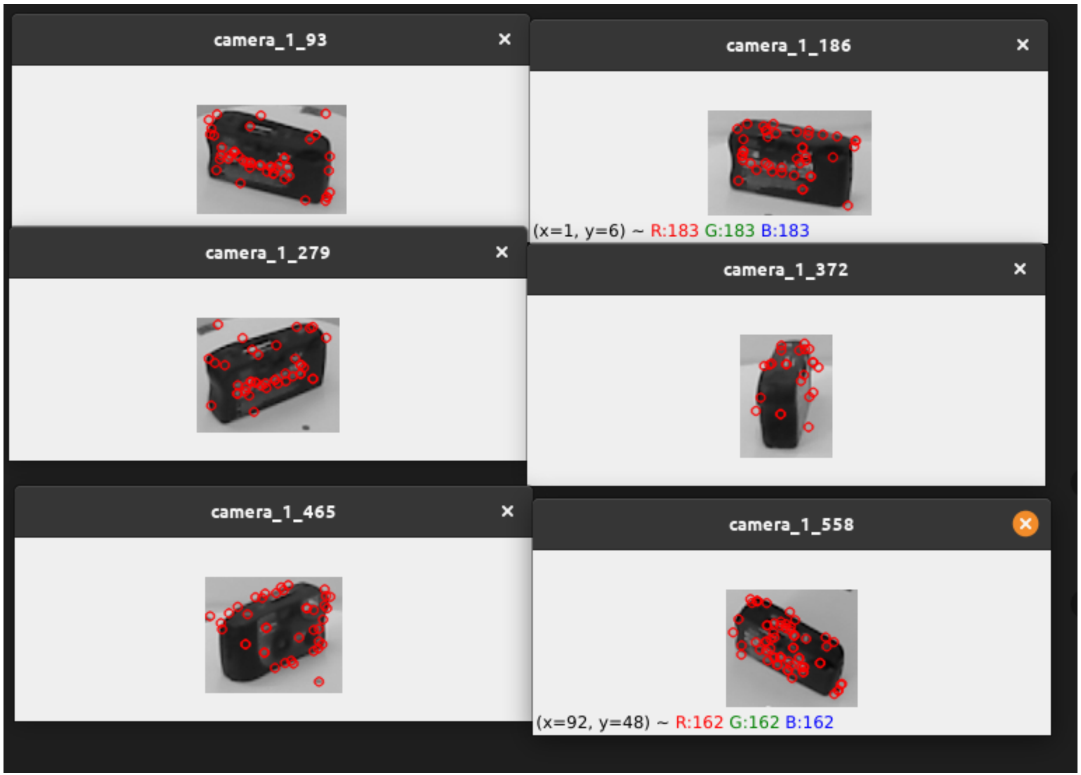


(x=13, y=8) ~ R:174 G:174 B:174

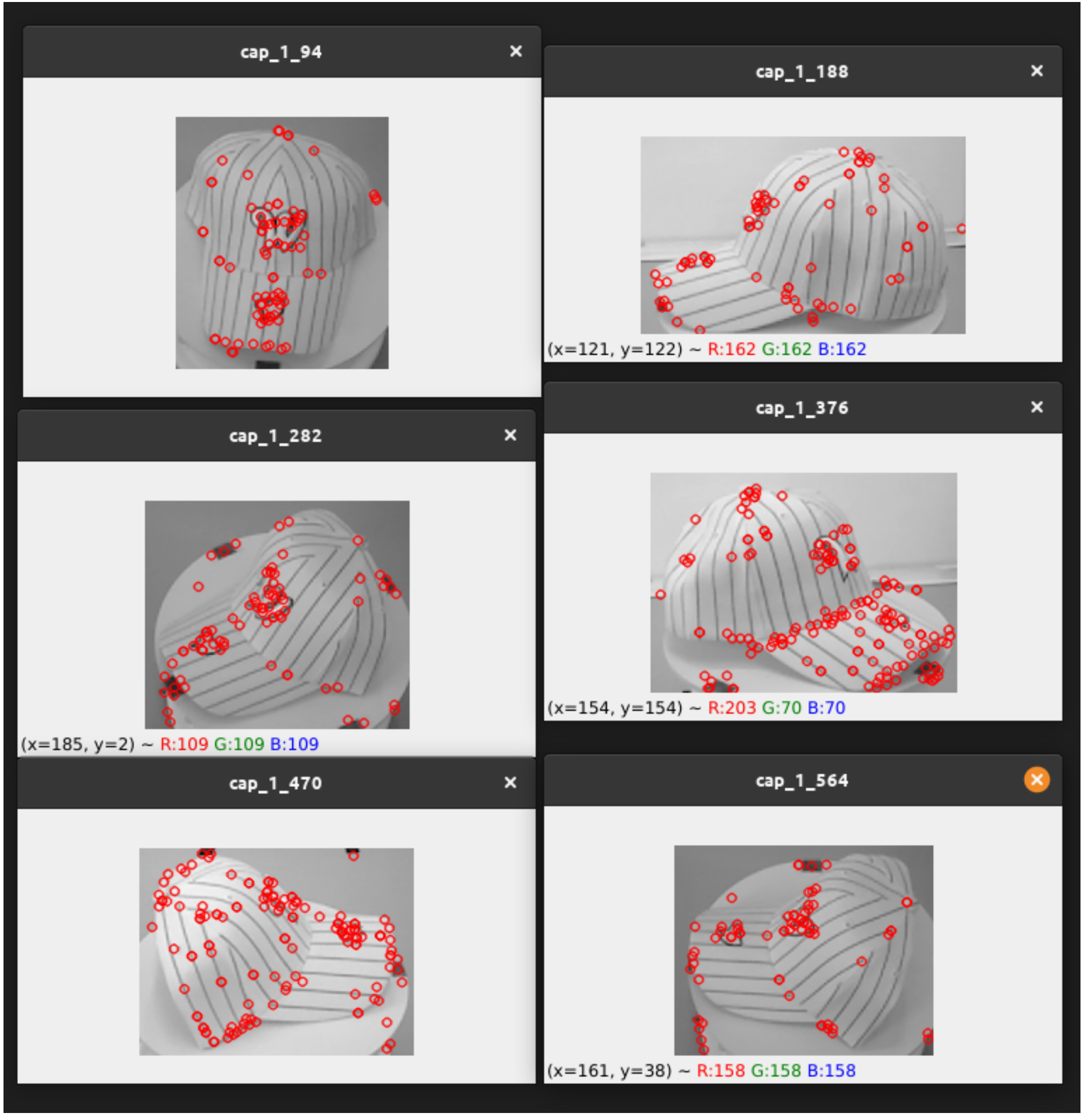
(x=2, y=22) ~ R:130 G:130 B:130



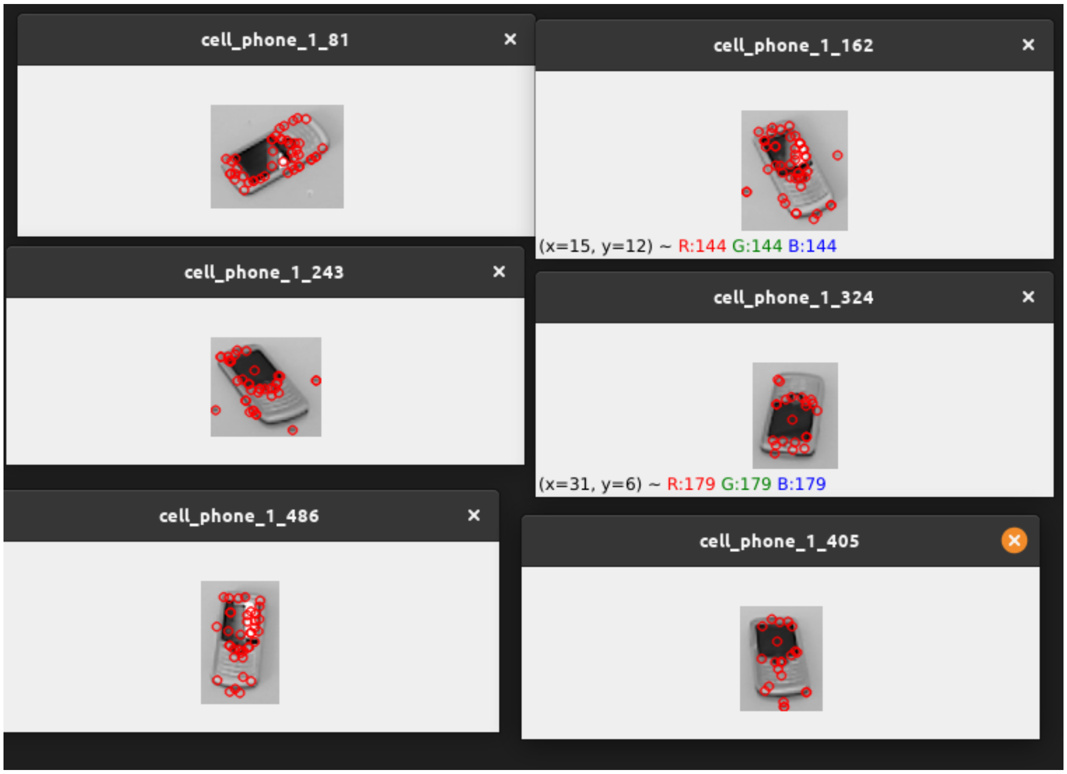
(x=0, y=12) ~ R:208 G:208 B:208



```
(x=1, y=6) ~ R:183 G:183 B:183
(x=92, y=48) ~ R:162 G:162 B:162
```



```
(x=121, y=122) ~ R:162 G:162 B:162
(x=185, y=2) ~ R:109 G:109 B:109
(x=154, y=154) ~ R:203 G:70 B:70
(x=161, y=38) ~ R:158 G:158 B:158
```



(x=15, y=12) ~ R:144 G:144 B:144

(x=31, y=6) ~ R:179 G:179 B:179

Object Recognition Algorithms

All of the algorithms below take in 2 parameters(You can also choose ORB based algorithms' edgeThreshold value but it has a default value). A training data set and a test data set. Test data set is chosen as the random 10% of all the data and the training data set is the 90% left. The match referred as KNN match is Brute Force Matcher KNN Match. All of the algorithms below follow the general routine which is:

1. Find keypoints and create descriptors from the given training data set using a feature detector and descriptor.

2. For every image in the test data set:

3. Find keypoints and create descriptors using the feature detector and descriptor used in training

4. Match the found descriptors with training descriptors using a matcher. Find the max number of matches for each object class.

5. Select the object class with the biggest max number of matches as the classification for the test object.

SIFT only

1. Find SIFT keypoints for each image in each object class in the training data set, create SIFT descriptors from them and relate the descriptors with the corresponding object class.

2. For each image in the test data set:

3. Find SIFT descriptors using SIFT keypoints

4. Match the found descriptors with every descriptor related with each object class using brute force KNN match

5. Take the good matches by comparing the 2 matches KNN match gives. A good match is where the first match's distance is smaller than the 0.4 times the second match's distance.

6. If the number of good matches is bigger than the max match count of the object class that is being tested against, assign it as the related object class' max match count.

7. Compare the max match count of all object classes. Pick the object class with the maximum value of max match count as the classification for the current test image.

ORB only

1. Generate ORB with parameters: nfeatures=100 to limit the max number of features to speed up the algorithm and edgeThreshold=10 to prevent the case where ORB finds no features in the image because images have small resolutions while not making the algorithm get very slow. edgeThreshold=12 can be used to make the algorithm faster while bringing some decline in accuracy which will be discussed in the **Results** section

2. Find ORB keypoints for each image in each object class in the training data set, create ORB descriptors from them and relate the descriptors with the corresponding object class.

3. For each image in the test data set:

4. Find ORB descriptors using ORB keypoints

5. Match the found descriptors with every descriptor related with each object class using brute force match algorithm with Hamming distance(param=cv.NORM_HAMMING) and cross check.

6. If the number of matches is bigger than the max match count of the object class that is being tested against, assign it as the related object class' max match count.

7. Compare the max match count of all object classes. Pick the object class with the maximum value of max match count as the classification for the current test image.

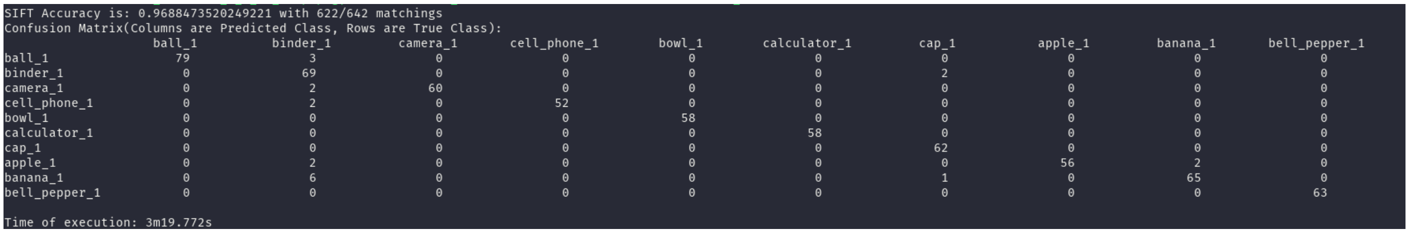
ORB with KNN Matching

1. Generate ORB with parameters: nfeatures=100 and edgeThreshold=10 or 12 just like the previous one.
2. Find ORB keypoints for each image in each object class in the training data set, create ORB descriptors from them and relate the descriptors with the corresponding object class.
3. For each image in the test data set:
4. Find SIFT descriptors using SIFT keypoints
5. Match the found descriptors with every descriptor related with each object class using brute force KNN match
6. Take the good matches by comparing the 2 matches KNN match gives. A good match is where the first match's distance is smaller than the 0.4 times the second match's distance.
7. If the number of good matches is bigger than the max match count of the object class that is being tested against, assign it as the related object class' max match count.
8. Compare the max match count of all object classes. Pick the object class with the maximum value of max match count as the classification for the current test image.

FAST with SIFT Descriptors

1. Find FAST keypoints for each image in each object class in the training data set, create SIFT descriptors from them and relate the descriptors with the corresponding object class.
2. For each image in the test data set:
3. Find SIFT descriptors using FAST keypoints
4. Match the found descriptors with every descriptor related with each object class using brute force KNN match
5. Take the good matches by comparing the 2 matches KNN match gives. A good match is where the first match's distance is smaller than the 0.4 times the second match's distance.
6. If the number of good matches is bigger than the max match count of the object class that is being tested against, assign it as the related object class' max match count.
7. Compare the max match count of all object classes. Pick the object class with the maximum value of max match count as the classification for the current test image.

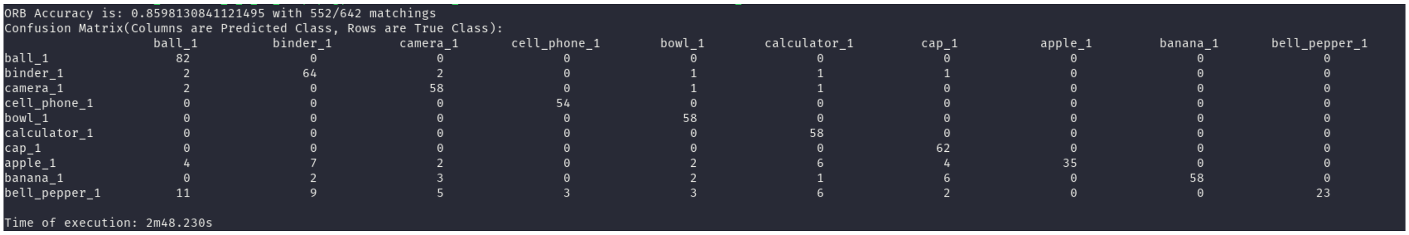
Results



```
SIFT Accuracy is: 0.9688473520249221 with 622/642 matchings
Confusion Matrix(Columns are Predicted Class, Rows are True Class):
                      ball_1       binder_1      camera_1    cell_phone_1     bowl_1     calculator_1     cap_1        apple_1       banana_1    bell_pepper_1
ball_1                  79            3             0             0             0             0             0             0             0             0
binder_1                0             69            0             0             0             0             2             0             0             0
camera_1                0             2             60            0             0             0             0             0             0             0
cell_phone_1            0             2             0             52            0             0             0             0             0             0
bowl_1                  0             0             0             0             58            0             0             0             0             0
calculator_1            0             0             0             0             0             58            0             0             0             0
cap_1                   0             0             0             0             0             0             62            0             0             0
apple_1                 0             2             0             0             0             0             0             56            2             0
banana_1                0             6             0             0             0             0             1             0             65            0
bell_pepper_1           0             0             0             0             0             0             0             0             0             63

Time of execution: 3m19.772s
```

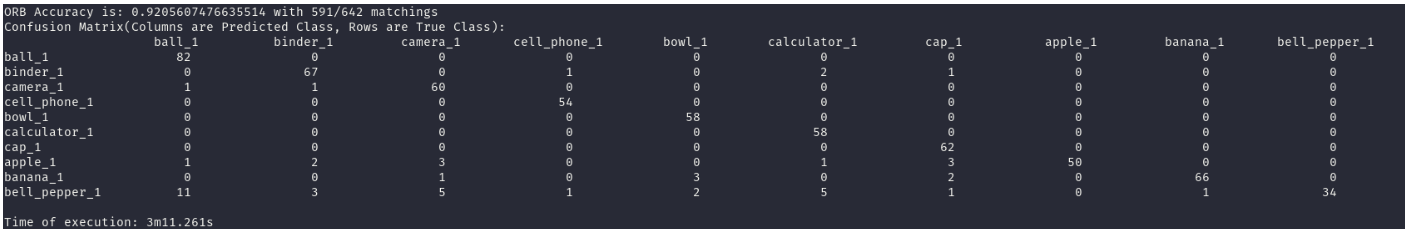
1. SIFT



```
ORB Accuracy is: 0.8598130841121495 with 552/642 matchings
Confusion Matrix(Columns are Predicted Class, Rows are True Class):
                      ball_1       binder_1      camera_1    cell_phone_1     bowl_1     calculator_1     cap_1        apple_1       banana_1    bell_pepper_1
ball_1                  82            0             0             0             0             0             0             0             0             0
binder_1                2             64            2             0             1             1             1             0             0             0
camera_1                2             0             58            0             1             1             0             0             0             0
cell_phone_1            0             0             0             54            0             0             0             0             0             0
bowl_1                  0             0             0             0             58            0             0             0             0             0
calculator_1            0             0             0             0             0             58            0             0             0             0
cap_1                   0             0             0             0             0             0             62            0             0             0
apple_1                 4             7             2             0             2             6             4             35            0             0
banana_1                0             2             3             0             2             1             6             0             58            0
bell_pepper_1           11            9             5             3             3             6             2             0             0             23

Time of execution: 2m48.230s
```

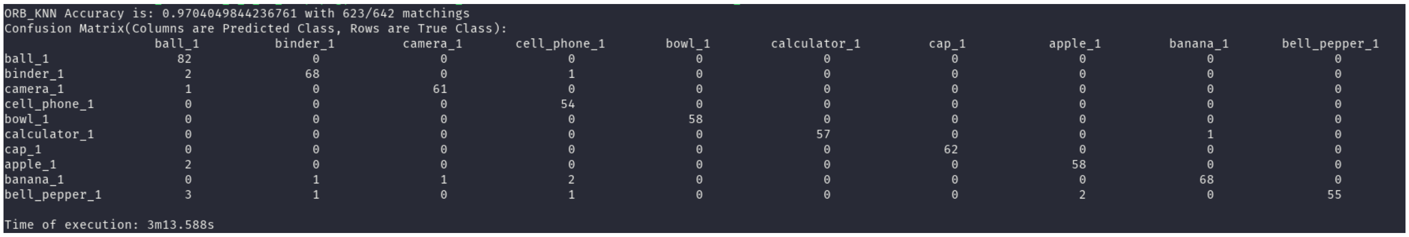
2. ORB with Edge Threshold=12



```
ORB Accuracy is: 0.9205607476635514 with 591/642 matchings
Confusion Matrix(Columns are Predicted Class, Rows are True Class):
                      ball_1       binder_1      camera_1    cell_phone_1     bowl_1     calculator_1     cap_1        apple_1       banana_1    bell_pepper_1
ball_1                  82            0             0             0             0             0             0             0             0             0
binder_1                0             67            0             1             0             2             1             0             0             0
camera_1                1             1             60            0             0             0             0             0             0             0
cell_phone_1            0             0             0             54            0             0             0             0             0             0
bowl_1                  0             0             0             0             58            0             0             0             0             0
calculator_1            0             0             0             0             0             58            0             0             0             0
cap_1                   0             0             0             0             0             0             62            0             0             0
apple_1                 1             2             3             0             0             1             3             50            0             0
banana_1                0             0             1             0             3             0             2             0             66            0
bell_pepper_1           11            3             5             1             2             5             1             0             1             34

Time of execution: 3m11.261s
```

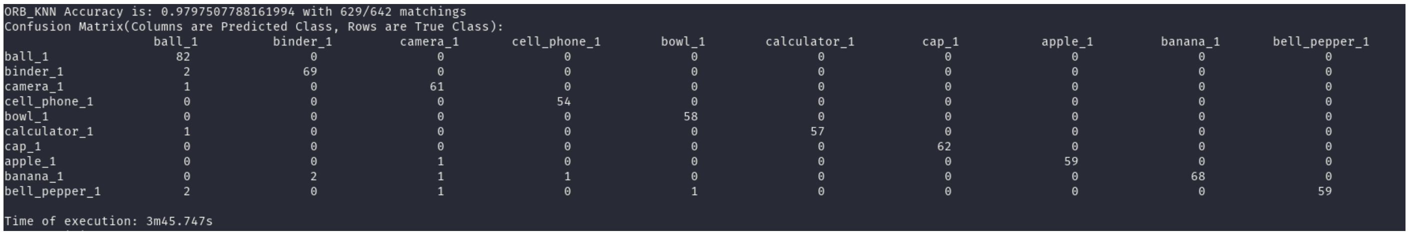
3. ORB with Edge Threshold=10



```
ORB_KNN Accuracy is: 0.9704049844236761 with 623/642 matchings
Confusion Matrix(Columns are Predicted Class, Rows are True Class):
                      ball_1       binder_1      camera_1    cell_phone_1     bowl_1     calculator_1     cap_1        apple_1       banana_1    bell_pepper_1
ball_1                  82            0             0             0             0             0             0             0             0             0
binder_1                2             68            0             1             0             0             0             0             0             0
camera_1                1             0             61            0             0             0             0             0             0             0
cell_phone_1            0             0             0             54            0             0             0             0             0             0
bowl_1                  0             0             0             0             58            0             0             0             0             0
calculator_1            0             0             0             0             0             57            0             0             1             0
cap_1                   0             0             0             0             0             0             62            0             0             0
apple_1                 2             0             0             0             0             0             0             58            0             0
banana_1                0             1             1             2             0             0             0             0             68            0
bell_pepper_1           3             1             0             1             0             0             0             2             0             55

Time of execution: 3m13.588s
```

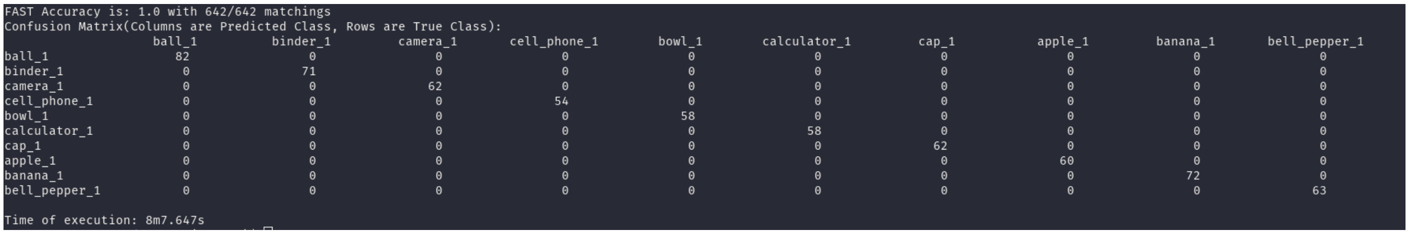
4. ORB+KNN matching with Edge Threshold=12



```
ORB_KNN Accuracy is: 0.9797507788161994 with 629/642 matchings
Confusion Matrix(Columns are Predicted Class, Rows are True Class):
                      ball_1       binder_1      camera_1    cell_phone_1     bowl_1     calculator_1     cap_1        apple_1       banana_1    bell_pepper_1
ball_1                  82            0             0             0             0             0             0             0             0             0
binder_1                2             69            0             0             0             0             0             0             0             0
camera_1                1             0             61            0             0             0             0             0             0             0
cell_phone_1            0             0             0             54            0             0             0             0             0             0
bowl_1                  0             0             0             0             58            0             0             0             0             0
calculator_1            1             0             0             0             0             57            0             0             0             0
cap_1                   0             0             0             0             0             0             62            0             0             0
apple_1                 0             0             1             0             0             0             0             59            0             0
banana_1                0             2             1             1             0             0             0             0             68            0
bell_pepper_1           2             0             1             0             1             0             0             0             0             59

Time of execution: 3m45.747s
```

5. ORB+KNN matching with Edge Threshold=10



```
FAST Accuracy is: 1.0 with 642/642 matchings
Confusion Matrix(Columns are Predicted Class, Rows are True Class):
                      ball_1       binder_1      camera_1    cell_phone_1     bowl_1     calculator_1     cap_1        apple_1       banana_1    bell_pepper_1
ball_1                  82            0             0             0             0             0             0             0             0             0
binder_1                0             71            0             0             0             0             0             0             0             0
camera_1                0             0             62            0             0             0             0             0             0             0
cell_phone_1            0             0             0             54            0             0             0             0             0             0
bowl_1                  0             0             0             0             58            0             0             0             0             0
calculator_1            0             0             0             0             0             58            0             0             0             0
cap_1                   0             0             0             0             0             0             62            0             0             0
apple_1                 0             0             0             0             0             0             0             60            0             0
banana_1                0             0             0             0             0             0             0             0             72            0
bell_pepper_1           0             0             0             0             0             0             0             0             0             63

Time of execution: 8m7.647s
```

6. FAST

All the execution times include the time it took to train the system. All the algorithms are tested using the same test data set.

ORB feature detector has a quirk called edge threshold. Usually key points at the edge of the images are not useful for most applications so ORB feature detector dismisses some area at the borders. Because our images are small, this causes ORB to miss a lot of features and sometimes not find any. So I experimented with this parameter with the aim to improve the accuracy of the algorithm while suffering from the slow down as little as possible. For both ORB algorithms I limited the ORB feature detector to 100 nfeatures because it didn't give me any accuracy boost in my tests but more than 100 slowed down the algorithm a lot.

Let's discuss the results above:

1. SIFT based algorithm have a really good accuracy with a decent execution time. Most false positives occurs in object with relatively low features. It seems high feature objects sometimes overrun the true class descriptors.

2. ORB based algorithm with Edge Threshold value of 12 gives us the fastest and least accurate algorithm(even though decent enough).

3. Decreasing the Edge Threshold value from 12 to 10 gives our algorithm better accuracy by about 6% but slows it down about 15%. I think it is not worth it to make this change as there are better options(such as SIFT and ORB+KNN) if we are looking for an algorithm that performs around this duration.

4. ORB with KNN matching is one of the better options that I mentioned above. While it is only 2 second slower than the ORB with edgeThreshold=10, it is 5% more accurate. There is no need to use the former when we have this algorithm.

5. Decreasing the Edge Threshold value from 12 to 10 didn't improve ORB with KNN matching algorithm as well. It only increases accuracy 1% while slowing the algorithm around 15%. I think this is the least efficient combo out of all.

6. For the last algorithm we have FAST with SIFT descriptors. Although it is the slowest out of all the algorithms, it has perfect accuracy. This result is not a coincidence, I tested it multiple times but didn't want to make report longer as it is long enough. If an algorithm with perfect or near perfect accuracy is needed, I would choose this one.

How the System Works

You can run the program by downloading the provided data set, extracting it into the same directory with main.py and run the main.py using Python. I am also submitting a version of the project with data set included.

I tested the terminal output in a bash terminal. Correct matches are printed green and wrong matches are printed wrong. If for any reason output is messed up, you can replace the colored_print function with the one specified in the source file by commenting them in and out.

The program works as follows:

1. All the subdirectories of data set is read and assigned as object classes.
2. All the files in these subdirectories are placed into a dictionary which uses object classes(parent directory names) as keys and files(images of objects) as values. This dictionary will be used as the training data set.
3. Random 10% of files in each object class is selected and put into an array which will be the test data set. Then these files are removed from the training data set.
4. Training data set is shuffled so that the files lose the order and spread across.
5. Each recognition algorithm explained in the previous section is applied to the data sets.
6. Information about the algorithm is collected while running; such as execution time, confusion matrix, number of correct matches.
7. The collected information is printed out to the terminal.