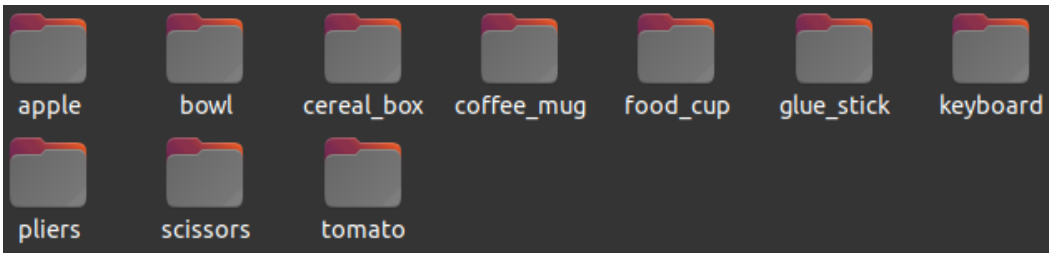
CSE463 – Introduction To Computer Vision

Homework #2 Report

Ömer Faruk Bitikçioğlu

161044010

Introduction



In this project we have 10 different objects and many pictures of them. Our main goal here is to train our program to detect these objects. For this reason we use feature detection techniques like SIFT and ORB. I tried to use SURF but it has a patent problem, I tried a few things but couldn't make it to use that algorithm.

We first load all of the images in a 2d matrix 'images[][]'. First index of the 2d matrix is for 10 object categories from 0 to 9. The second index is for the images under this object category.

FindDes method traverses all the images and find their descriptors, keeps them in a list as 2d list as the images itself.

FindID method takes an image, detect its features with a feature detection algorithm, compares its descriptor with 90% of the images' descriptors. Keeps the good matches in a list. Sums the count of good matches for each object category. In turn we have a list named matchList, shows the good match count for each object category with respect to given image input. Treshold input can be changed to get more correct results. According to my experiments value of 500 is enough. Also we can change the ratio when we find matches to eliminate failures. I experimented as 0.5 is ideal for this project.

When we try, it founds the keyboard and cereal box correctly. The other objects have little details and are hard to extract features, so they are not detected by the algorithm.

Features detected by ORB

The ORB Algorithm

ORB specifies the rBRIEF algorithm as follows:

1) Run each test against all training patches.
2) Order the tests by their distance from a mean of 0.5, forming the vector T.
3) Greedy search:
   - Put the first test into the result vector R and remove it from T.
   - Take the next test from T, and compare it against all tests in R. If its absolute correlation is greater than a threshold, discard it; else add it to R.
   - Repeat the previous step until there are 256 tests in R. If there are fewer than 256, raise the threshold and try again

rBRIEF shows significant improvement in the variance and correlation over steered BRIEF.

```
[1060, 0, 0, 0, 0, 0, 0, 3, 0, 0]
keyboard==keyboard
[822, 0, 0, 0, 0, 0, 0, 1, 0, 0]
keyboard==keyboard
[1002, 0, 0, 0, 0, 0, 0, 3, 0, 0]
keyboard==keyboard
[827, 0, 1, 0, 0, 0, 0, 1, 0, 0]
keyboard==keyboard
[1190, 0, 0, 0, 0, 0, 0, 1, 0, 0]
keyboard==keyboard
[1227, 0, 0, 0, 0, 0, 0, 1, 0, 0]
keyboard==keyboard
[590, 0, 0, 0, 0, 0, 0, 1, 0, 0]
keyboard==keyboard
[513, 0, 0, 0, 0, 0, 0, 0, 0, 0]
keyboard==keyboard
[986, 0, 0, 0, 0, 0, 0, 0, 0, 0]
keyboard==keyboard
[1166, 0, 1, 0, 0, 0, 0, 3, 0, 0]
keyboard==keyboard
[705, 0, 0, 0, 0, 0, 0, 1, 0, 0]
keyboard==keyboard
[829, 0, 2, 0, 0, 0, 0, 0, 0, 0]
keyboard==keyboard
[772, 0, 2, 0, 0, 0, 0, 1, 0, 0]
keyboard==keyboard
[637, 0, 0, 0, 0, 0, 0, 2, 0, 0]
keyboard==keyboard
[1271, 0, 0, 0, 0, 0, 0, 5, 0, 0]
keyboard==keyboard
[916, 0, 1, 0, 0, 0, 0, 0, 0, 0]
keyboard==keyboard
[957, 0, 0, 0, 0, 0, 0, 0, 0, 0]
keyboard==keyboard
[589, 0, 1, 0, 0, 0, 0, 5, 0, 0]
keyboard==keyboard
[564, 0, 0, 0, 0, 0, 0, 0, 5, 0]
keyboard==keyboard
[1024, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0]
keyboard==keyboard
[1209, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0]
keyboard==keyboard
[935, 0, 0, 0, 0, 0, 0, 0, 0, 0]
keyboard==keyboard
[933, 0, 0, 0, 0, 0, 0, 1, 0, 0]
keyboard==keyboard
[745, 0, 0, 0, 0, 0, 0, 0, 0, 0]
keyboard==keyboard
```

```
[806, 0, 0, 0, 0, 0, 0, 1, 0, 0]
keyboard==keyboard
[1033, 0, 0, 0, 0, 0, 0, 1, 0, 0]
keyboard==keyboard
[883, 0, 0, 0, 0, 0, 0, 0, 0, 0]
keyboard==keyboard
[1036, 0, 0, 0, 0, 0, 0, 0, 0, 0]
keyboard==keyboard
[968, 0, 0, 0, 0, 0, 0, 3, 0, 0]
keyboard==keyboard
[884, 0, 1, 0, 0, 1, 0, 12, 0, 0]
keyboard==keyboard
[535, 0, 0, 0, 0, 0, 0, 2, 0, 0]
keyboard==keyboard
[4, 0, 0, 0, 0, 0, 0, 1730, 0, 0]
cereal_box==cereal_box
[1, 0, 0, 0, 0, 0, 0, 1117, 0, 0]
cereal_box==cereal_box
[2, 0, 2, 0, 0, 0, 0, 997, 1, 0]
cereal_box==cereal_box
[0, 0, 0, 0, 0, 0, 0, 766, 0, 0]
cereal_box==cereal_box
[1, 0, 0, 0, 0, 0, 0, 1286, 0, 0]
cereal_box==cereal_box
[1, 0, 0, 0, 0, 0, 0, 1679, 0, 0]
cereal_box==cereal_box
[2, 0, 0, 0, 0, 0, 0, 1387, 0, 0]
cereal_box==cereal_box
[2, 0, 0, 0, 0, 0, 0, 1482, 0, 0]
cereal_box==cereal_box
[0, 0, 0, 0, 0, 0, 0, 1346, 0, 0]
cereal_box==cereal_box
[1, 0, 0, 0, 0, 0, 0, 652, 0, 0]
cereal_box==cereal_box
[1, 0, 0, 0, 0, 0, 0, 1032, 0, 0]
cereal_box==cereal_box
[3, 0, 0, 0, 0, 0, 0, 567, 0, 0]
cereal_box==cereal_box
[2, 0, 0, 0, 0, 0, 0, 1852, 1, 0]
cereal_box==cereal_box
[0, 1, 0, 0, 0, 1, 0, 844, 0, 0]
cereal_box==cereal_box
[0, 0, 0, 0, 0, 0, 0, 587, 0, 0]
cereal_box==cereal_box
[1, 0, 0, 0, 0, 0, 0, 989, 0, 0]
cereal_box==cereal_box
[6, 1, 0, 0, 0, 0, 0, 1897, 0, 0]
cereal_box==cereal_box
[1, 0, 0, 0, 0, 0, 0, 1008, 0, 0]
cereal_box==cereal_box
```

```
[3, 0, 0, 0, 0, 0, 0, 1201, 0, 0]
cereal_box==cereal_box
[1, 0, 0, 0, 0, 0, 0, 1501, 0, 0]
cereal_box==cereal_box
[0, 0, 0, 0, 0, 0, 0, 1053, 0, 0]
cereal_box==cereal_box
[1, 0, 0, 0, 0, 1, 0, 1875, 0, 0]
cereal_box==cereal_box
[9, 0, 3, 0, 0, 0, 0, 1814, 0, 0]
cereal_box==cereal_box
[2, 0, 0, 0, 0, 0, 0, 1235, 0, 0]
cereal_box==cereal_box
[0, 0, 0, 0, 0, 0, 0, 1260, 0, 0]
cereal_box==cereal_box
[1, 0, 0, 0, 0, 0, 0, 1069, 0, 0]
cereal_box==cereal_box
[0, 0, 0, 0, 0, 0, 0, 583, 0, 0]
cereal_box==cereal_box
[0, 0, 0, 0, 0, 1, 0, 1120, 0, 0]
cereal_box==cereal_box
[0, 0, 0, 0, 0, 0, 0, 1229, 0, 0]
cereal_box==cereal_box
[1, 0, 0, 0, 0, 0, 0, 1393, 0, 0]
cereal_box==cereal_box
[7, 0, 2, 0, 0, 1, 0, 631, 0, 0]
cereal_box==cereal_box
[1, 0, 0, 0, 0, 0, 0, 892, 0, 0]
cereal_box==cereal_box
```

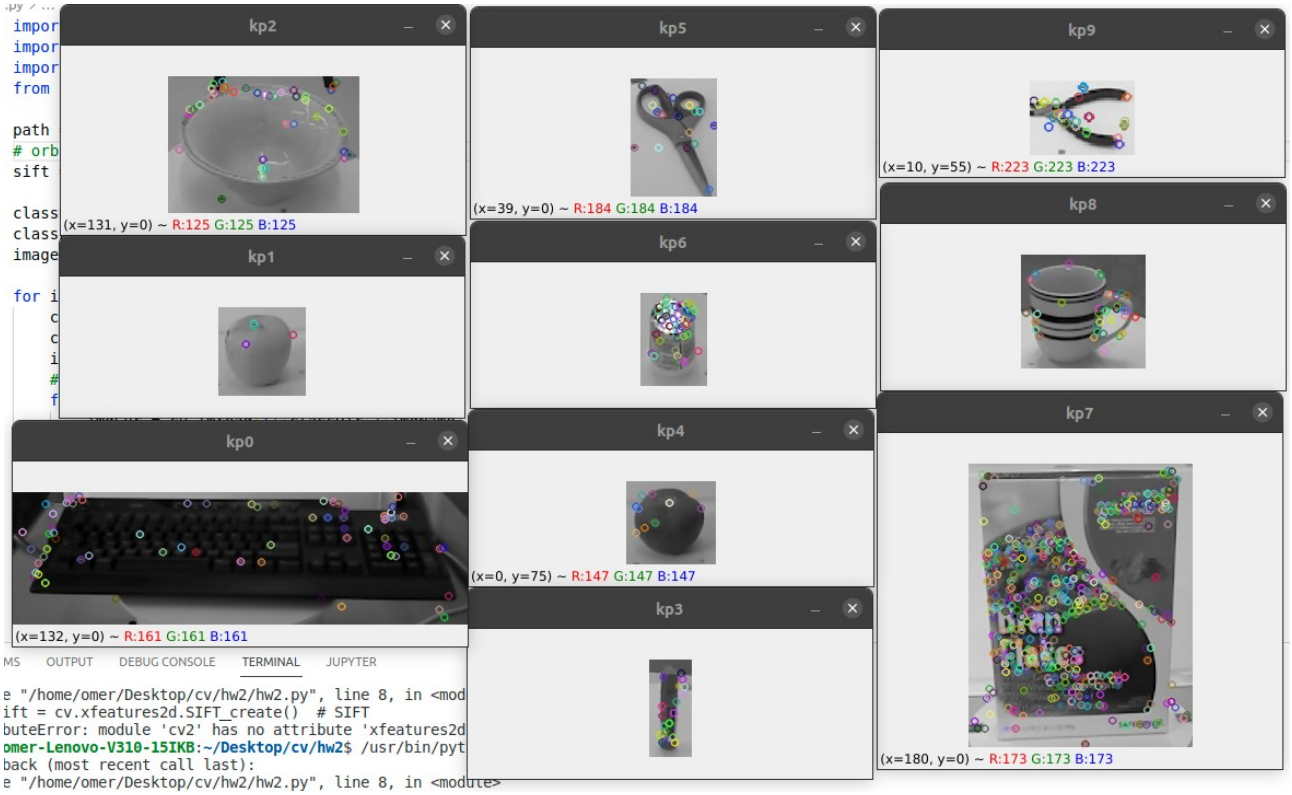
The arrays shown in the results are total good match of the given input with respect to the 10 object categories.

Confusion Matrix

When we use ORB algorithm with 0.75 ratio we have a confusion matrix as below:

```
[[ 0  0  2  0  0  3  0  0]
 [ 0  0 35  0  0 18  0  0]
 [ 0  0 57  0  0  0  0  0]
 [ 0  0 53  2  0 18  0  0]
 [ 0  0 29  0  0  7  0  0]
 [ 0  0  0  0  0 52  0  0]
 [ 0  0 31  0  0  6  0  0]
 [ 0  0  1  0  0  0  0  0]]
```

Features detected by SIFT



The SIFT Algorithm

SIFT is quite an involved algorithm. There are mainly four steps involved in the SIFT algorithm. We will see them one-by-one.

- Scale-space peak selection: Potential location for finding features.
- Keypoint Localization: Accurately locating the feature keypoints.
- Orientation Assignment: Assigning orientation to keypoints.
- Keypoint descriptor: Describing the keypoints as a high dimensional vector.
- Keypoint Matching

Confusion Matrix

When we use SIFT algorithm with 0.75 ratio we have a confusion matrix as below:

```
[[ 0  1 19  0  0  0 37  0  0  0]
 [ 0  7  9  0  0  0 42  0  0  0]
 [ 0  0 56  0  0  0  1  0  0  0]
 [ 0  0 30 15  0  0 33  0  0  0]
 [ 0  0  9  0 20  0 52  0  0  0]
 [ 0  0 19  0  0  4 56  0  0  0]
 [ 0  0  0  0  0  0 52  0  0  0]
 [ 0  0 13  0  0  0 28 16  0  0]
 [ 0  0 19  0  0  0 34  0  9  0]
 [ 0  0  7  0  0  0 62  0  0  1]]
```

With SIFT, we extracted more features and detected more objects.
The printed results were very long and I didn’t wanted to add it here. When the program executed, they can be seen