Mateusz Wygoda
211872

# *Methods of Computational Intelligence and Decision Making*

Topic: Ability comparison of algorithms SIFT, SURF, BRIEF, ORB in images matching.

## 1. Introduction

Feature detection is the process of computing the abstraction of the image information and making a local decision at every image point to see if there is an image feature of the given type existing in that point. I prepared small experiment to test abilities of following:
SIFT, SURF, BRIEF, ORB. Worth to say that, expect ORB which is openCV lab algorithm, all algorithms are patented and paid for commercial usage (free for academic).

## 2. Main goal

Main goal of the experiment was to find matches between two images cola images from internet and my own picture taken using my phone (image 2). Bottle of cola was in different environment, light, rotation and a little bit worse quality.

## 3. Experiment description

Using Python and OpenCV I implement simple program with following steps:
Create Detectors, input images to detector, obtain key points and compare key points from both images. To find matches I used brute force matcher (BFMatcher) from OpenCV library.

Comparison was in few criteria:

- time to obtain key points
- Key points found
- count of “good” matches
- Visual matches comparison

4. Experiment results

4.1 Time to obtain key points

In this part of experiment average time of feature extraction was measured like for example:

```
orb = cv2.ORB_create(nfeatures=1500)
start = time.time()
keypoints, descriptors = orb.detectAndCompute(img, None)
stop = time.time()

print("time", stop - start)
```

At this point concrete time doesn't matter but what is worth to observe is the fact that SIFT and SURF algorithms are much slower than BRIEF and ORB.

| | time[s] |
|---|---|
| SIFT | 0,11 |
| SURF | 0,22 |
| BRIEF | 0,0088 |
| ORB | 0,035 |

*Table 1 Algorithms execution time*

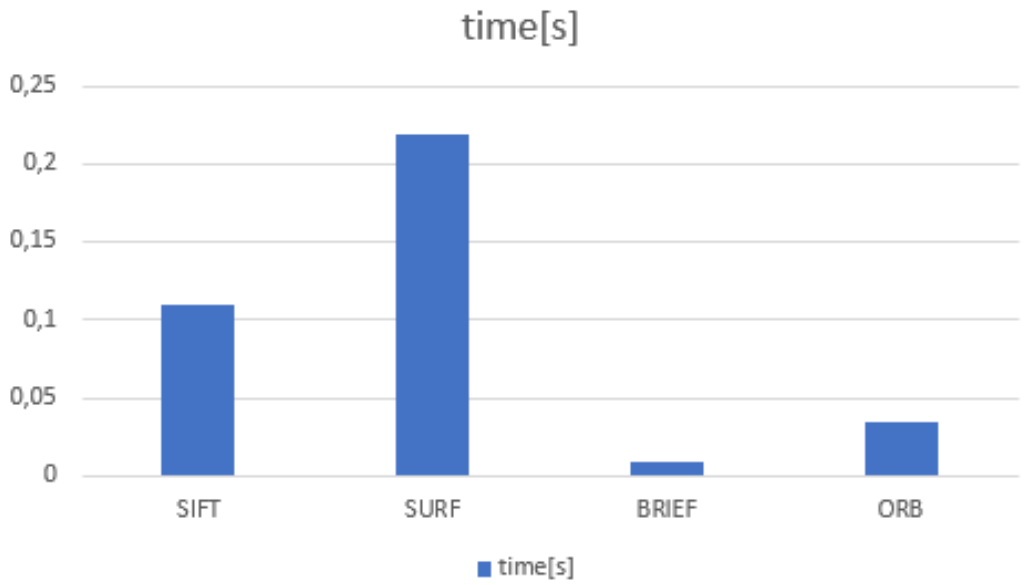


*Figure 1 Time chart*

4.2 Key points found

In this step every key point extracted like was counted as following:

```
keypoints, descriptors = orb.detectAndCompute(img, None)
print("Count:", len(keypoints))
```

We can observe that Brief detected the least key points, but it doesn't mean that this algorithm has the worst accuracy. ORB algorithm with default constructor is set to extract only 500 key points, if we set it to maximum (nfeatures=1500) there will appear 1453 key points.

| | keypoints |
|---|---|
| SIFT | 247 |
| SURF | 861 |
| BRIEF | 106 |
| ORB | 500 |

*Table 2 keypoints found*

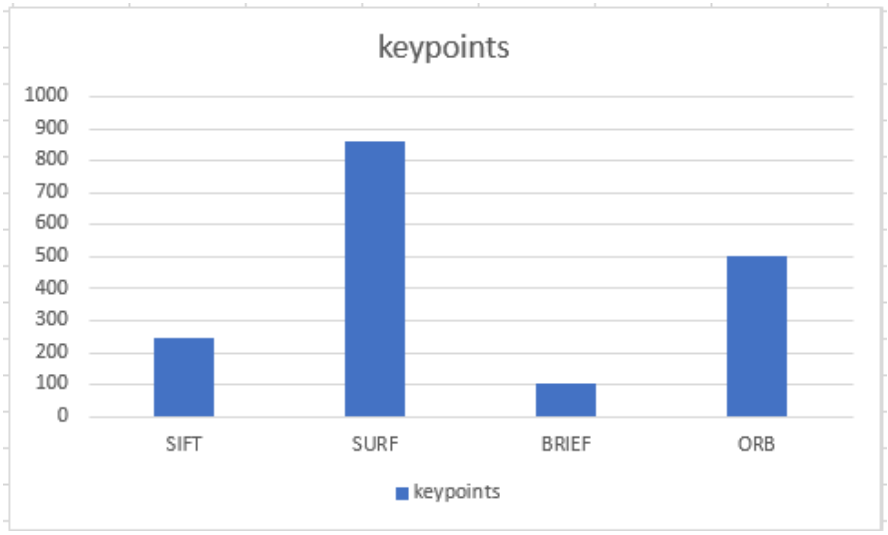


*Figure 2 keypoints chart*

4.3 “Good” matches

Matches was searched by brute force, but I used method knnMatch() like

```
matches = bf.knnMatch(des1, des2,k=2)
```

because I wanted to recive match.distance for features from both images and then classify base on condition:

```
for m,n in matches:
    if m.distance < 0.80*n.distance:
        good.append([m])
```

as good match. It wasn’t the most accurate criteria but I decided to compere in the next step fewer, somehow better matches.

| | good matches |
|---|---|
| SIFT | 61 |
| SURF | 71 |
| BRIEF | 7 |
| ORB | 21 |

*Table 3 good matches*

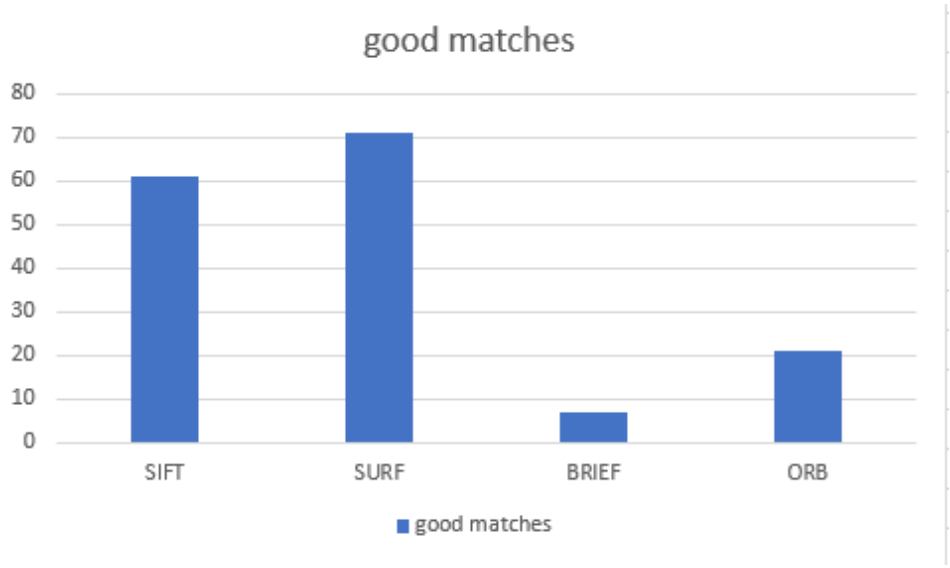


*Figure 3 good matches chart*

## 4.4 Visual matches comparison

### 4.4.1 SIFT

*Figure 4 SIFT Matching Result*

### 4.4.2 SURF

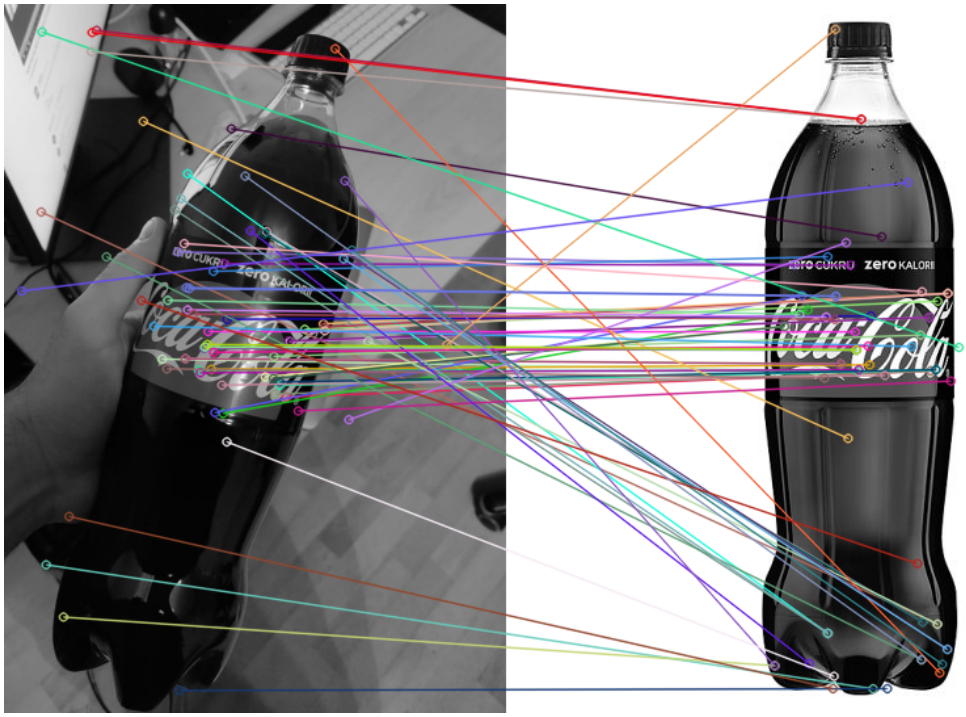

*Figure 5 SURF Matching Result*

### 4.4.3 BRIEF

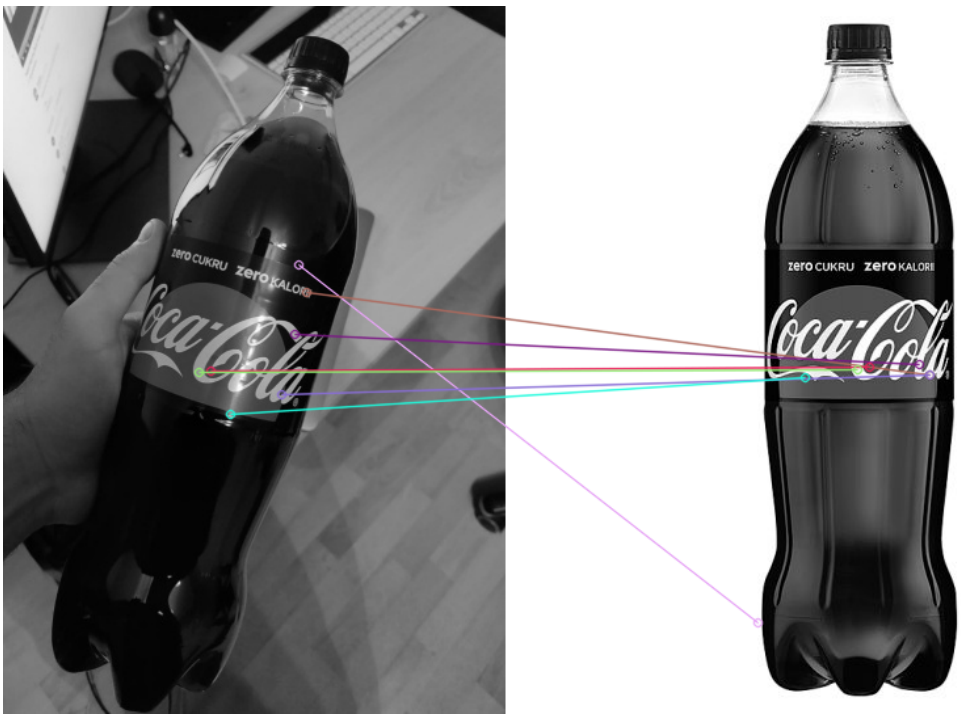


*Figure 6 BRIEF Matching Result*

4.4.4 ORB

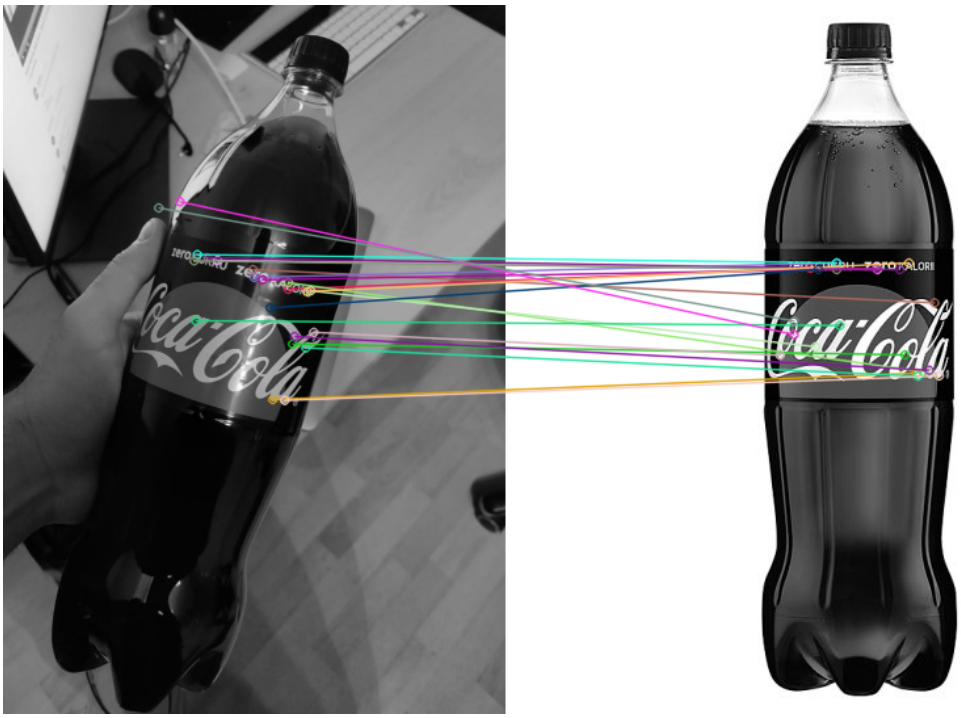

*Figure 7 ORB Matching Result*

5. Summary

The most important step was visual comparison. Based on that we can see that both SIFT and SURF get the most matches and those matches contains a lot of correct matches. Only disadvantage of those methods is that both SIFT and SURF are computational expensive and time consuming. BRIEF on that example has the worse matching results, but is the fastest. ORB presenting the best accuracy of matching. We can observe that most of matches are correct and algorithm itself is fast moreover this algorithm is free to commercial use.

6. References

[1] https://docs.opencv.org/3.4
[2] http://www.willowgarage.com/sites/default/files/orb_final.pdf
[3] https://arxiv.org/pdf/1710.02726.pdf