Report Image Features and Matching

EQ2425, Project 1

Chierchia Remi
remich@kth.se

Castellan Sebastiano
sebcas@kth.se

September 17, 2021

Summary

In this project, the goal is to analyze two different algorithms for extract feature key points. These two are SIFT (Scale Invariant Feature Transform) and SURF (Speeded-Up Robust Features). For comparing we have two main ways. The first one is analyzing the different responses that the descriptor has with an altered image, and compare with the descriptors of the original image through computing the repeatability. The two ways to modify the image are rotation and scale. We found that SIFT works better on average but SURF has the best result for 90 angles. The second way we implement different matching algorithms and after finding one that performs the best matching, we use it to differentiate the results for SIFT and SURF. These matching algorithms are fixed threshold, nearest neighbor, and nearest neighbor distance ratio. The last one proves more accurate matching and the comparison between SIFT and SURF shows that SIFT is again the best algorithm. The conclusions explain every founds of this project emphasizing that on our condition SIFT work better on average for all our tests meanwhile SURF has a comparable behavior but using fewer resources and less computational time.

1 Introduction

The goal of this project is to analyze the algorithms SIFT [4] and SURF [2] [1] and test the "repeatability" measures against rotation and scaling of the images. The cited algorithms exploit and analyze the "features" of an image, that describes the characteristics of the visualized scenario.

SIFT stands for Scale Invariant Feature Transform, meaning that it is robust against manipulations such illumination, scale, perspective, and occlusions. These features are discriminated for location, orientation, and scale, based on a comparison of Gaussian filtered and scaled images. Moreover, it is known to be a real-time technique. More information can be found in the official article [4].

SURF stands for Speeded-Up Robust Features, and as the name suggests it is a quicker version of SIFT algorithm. It is based on the "Integral Image" concept, which is an intermediate image representation allowing fast computation of rectangular features at a constant speed. The rectangular feature considers the height and width property of the objects in the grayscale image, to evaluate a particular point in an image. An important difference with respect to SIFT method, is that here is up-sampled the filter instead of down-sampled the image. The Haar wavelet transform is so applied to get the horizontal, vertical and diagonal components. For further information rely on [2] and [1].

In this project we used Matlab, the library VLFeat [3] for SIFT algorithm, and the Matlab function "detectSURFFeatures" for SURF feature detection.

The VLFeat library requires the input image and returns a feature vector containing the position of the detected keypoints, and a descriptor matrix. This matrix contains in each column the 128 elements used by SIFT algorithm to determine the position, scale, and orientation of such detections.

"detectSURFFeatures()", instead, given an input image, returns a SURFPoints object, containing information about SURF features detected in the 2D grayscale input image I. The function "extractFeatures()", requires the SURFPoints object and the input image, and returns a feature matrix similar to the descriptor matrix of SIFT but with 64 rows, and a descriptor object.

2 Problem Description

The project is divided into two main sections. The first part is about the robustness of the keypoints detectors. SIFT and SURF are applied to a reference image. In order to visualize only a few hundred features, to be able to visually examine the performances, thresholding parameters are necessary to be set. For the SIFT case, edge and peak thresholds are chosen, while for SURF we simply chose the strongest ones. Then, the input image is firstly rotated every 15 degrees, from 0 to 360, and then is scaled of factors $m^0, ..., m^8$, where $m = 1.2$. In both cases, the two algorithms are compared to determine which one performs better.

The second part is about feature matching. Here two images representing the same scenario, but from different perspectives, are compared. The matching algorithms implemented are three, "fixed threshold", "nearest neighbor", and "nearest neighbor distance ratio". For the first and the latter algorithms the best and the sub-optimal results are compared, and each one is evaluated with SIFT. The last point of this section is to apply the "nearest neighbor distance ratio" algorithm using SURF, comparing the result with the SIFT implementation.

The repeatability measure is defined as the number of matching features between the original image and the modified images, divided by the number of detected features in the original image. To compute the repeatability, we predict the positions $[x', y']$ where the keypoints should ideally appear in the modified images for both cases, rotation and scale. If these keypoints are not present within a certain neighborhood in both images, the repeatability measure is decreased. To allow some approximations, we search for a nearby keypoint which is detected in the modified image with coordinate $[x_0, y_0]$, satisfying $|x_0 - x'| \leq 2$ and $|y_0 - y'| \leq 2$. If such a condition has verified a point $[x_0, y_0]$ is found, and the "number of feature matches" is incremented by 1.

3 Results

3.1 Robustness of Keypoint Detector

3.1.1 a

The goal of this section is to display SIFT and SURF keypoint detectors over the image 1.

Figure 1: In this figure is represented the reference image in RGB space.

In order to being able to visually examine the comparison, we selected few hundreds of features. To do that we inserted the values of edge and peak threshold directrly in the vl_sift (1) function, with value respectively of 10 and 15.

$$[f1, d] = vl\_sift(I1a, 'edgethresh', 10, 'peakthresh', 15); \tag{1}$$

Similarly for the SURF algorithm we selected the strongest values of the same amount of the detected with SIFT method, so in this case 227 (2).

$$\begin{aligned} points &= detectSURFFeatures(I2a); \\ points &= points.selectStrongest(size(f1, 2)); \end{aligned} \tag{2}$$

The result can be seen in figure 2.

Figure 2: On the left are represented the SIFT feature, while on the right are SURF feature detected.

In the above image we can see how the detections are sensible on the edges and corners especially in the roof and on the KTH logo. SIFT resulted to be more sensible on the windows in the left of the image, while SURF is not. This could be due to the selection of the strongest feature in the SURF algorithm, but we denoted that even augmenting the number of total features to display (changing peak threshold to 13), the result is similar (3).

Figure 3: We can see how incrementing the feature displayed (510) the result is consistent.

3.1.2 b

In this section, we evaluated the rotation repeatability versus rotation angle in increment of 15 degrees from 0 to 360 for both the algorithms. The difficult part is to calculate where the next point should be in the resulted rotated image, to compare with the detected one. We can see in the plots below (4)(5) how SURF is invariant every 90 degrees, while SIFT has a different pattern. Moreover, in general SIFT algorithm performs better overall angles, while SURF detects only 65% of the features at 150 degrees.

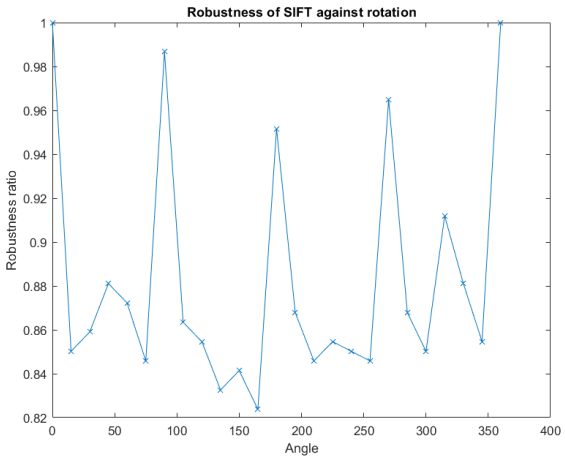

Figure 4: Repeatability of SIFT against rotation.

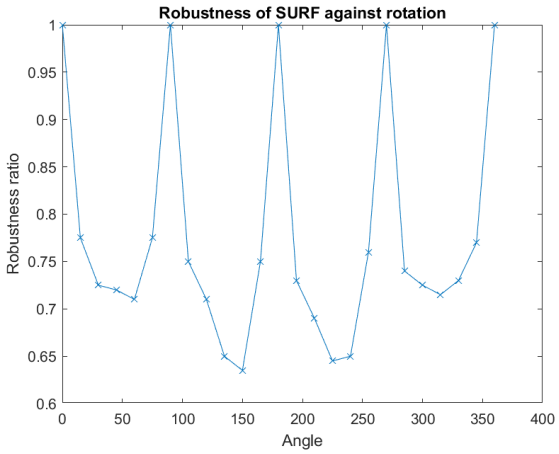

Figure 5: Repeatability of SURF against rotation.

3.1.3 c

Here we evaluated the repeatability of the two algorithms against scaling. Similarly than for the rotation, we scale the original points and then we compare them with the detected ones (6)(7). In the SIFT plot we observe how the behavior seems to be linear, with a maximum degradation of 60% when the scaling factor is 4.3. In the SURF plot, instead, we observe a behavior more sensible to scaling, with an intense degradation since the very first iterations.

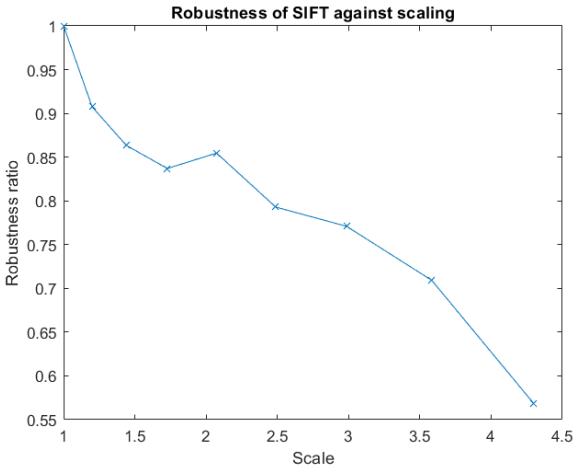

Figure 6: Repeatability of SIFT against scaling.

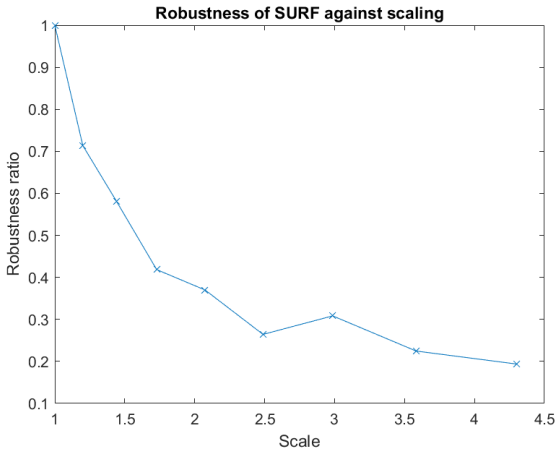

Figure 7: Repeatability of SURF against scaling.

3.2 Image Feature Matching

3.2.1 a

In this section we show how SIFT algorithm detected the features in 2 different images (8). Being the same scenario but from a different perspective, and keeping the threshold values as in the sections above, we obtain 177 features for the image on the left, and 227 features on the right image that is the same as before.

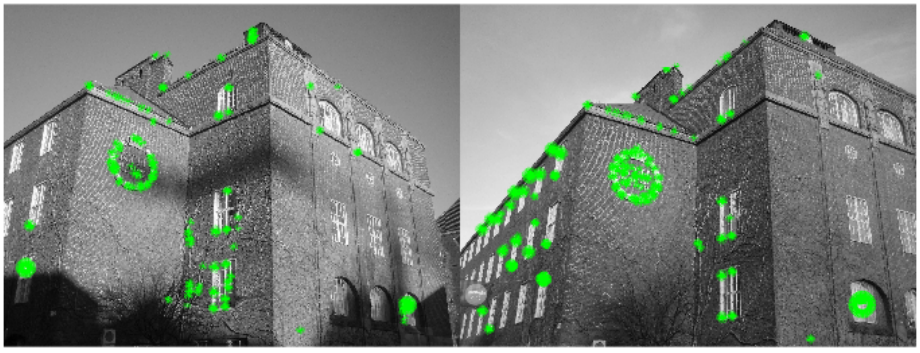

Figure 8: SIFT features on 2 different images.

3.2.2 b

In this section, we implemented the algorithm "fixed threshold", that compute the distance between features keypoints with the following function.

```
for i = 1 : n
    for j = 1 : m
        distance(j,i) = sqrt(sum((d1(:,i)-d2(:,j)).^2));
    end
end
```

Using the descriptor matrix d, we iterate through each column and measure the distance between each feature vector of 128 elements of each image. The function evaluates the simple euclidean distance. Then, the smaller distance is taken as the probable match and compared with the set threshold.

```
Id = v1<=threshold;
```

If the current distance is more than the threshold, the pair of keypoints are discarded. For the best result we set the threshold to 60, obtaining the result in figure (9).

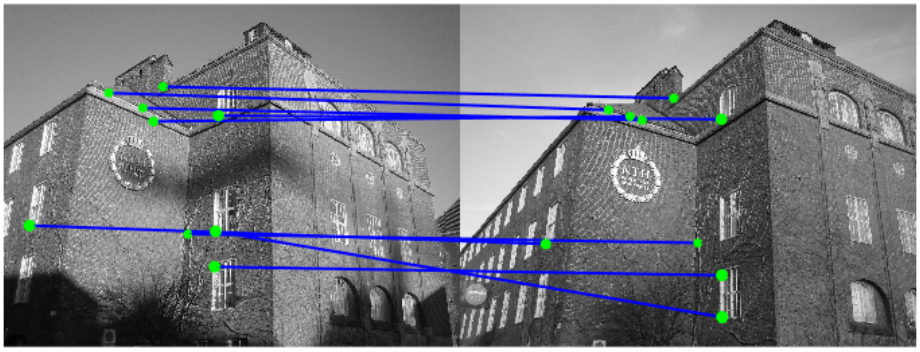

Figure 9: SIFT fixed threshold feature matching (threshold=60).

For the sub-optimal result, we choose a threshold equal 70 (10).

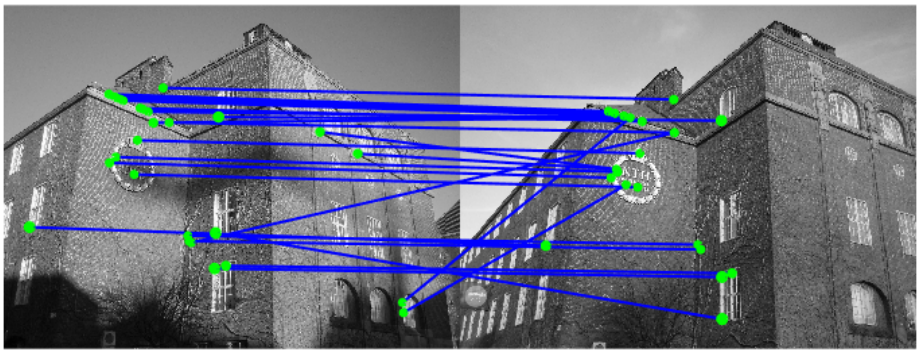

Figure 10: SIFT fixed threshold feature matching (threshold=70).

We observe that the best result still misses some matches, that is what we expected, given that is chosen the point that is below the threshold, so is not checked if a point actually belongs to it. In the sub-optimal case, we see how this behavior is highlighted.

3.2.3 c

In this section we evaluate the algorithm "nearest neighbor" (11). We choose to display 30 matches, and we clearly see an improvement with respect to the "fixed threshold". We still founded pairs of keypoints with a wrong matching due to the association with the nearest point, without any further assumption.

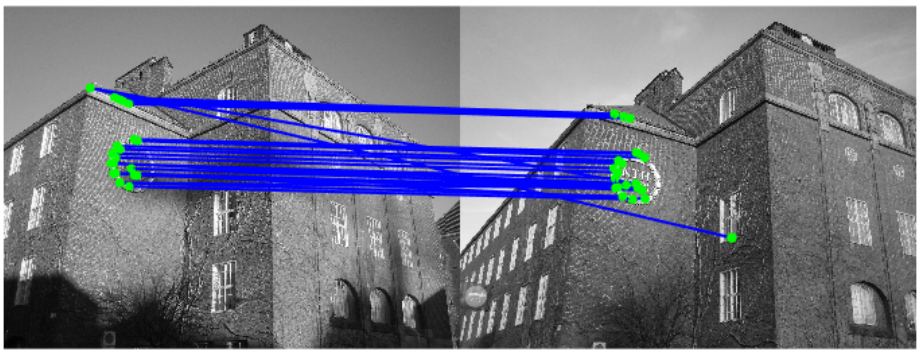

Figure 11: SIFT nearest neighbor feature matching (first 30 matches).

3.2.4 d

The last algorithm is the "nearest neighbor distance ratio". Here the nearest neighbor is divided to the second nearest neighbor in order to compute the ratio.

```
criteria = v1./v2;
Id = criteria<=ratio;
```

As for the previous cases, if the current criteria is less that a chosen threshold, the keypoint is considered valid. In the figure (12) we can observe how the algorithm is optimal with a ratio equal to 0.82.

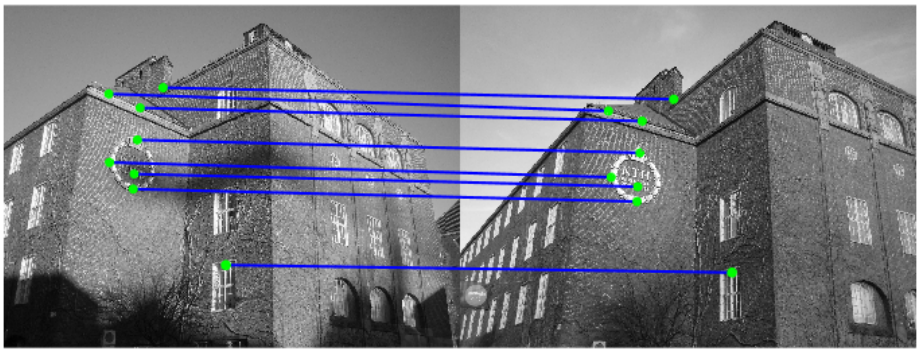

Figure 12: SIFT nearest neighbor distance ratio feature matching (ratio=0.82).

For the sub-optimal result, we choose a ratio equal to 0.9, where we start seeing mismatches (13). The error is still low compared to the true matches, so we can say that this algorithm is the best so far.

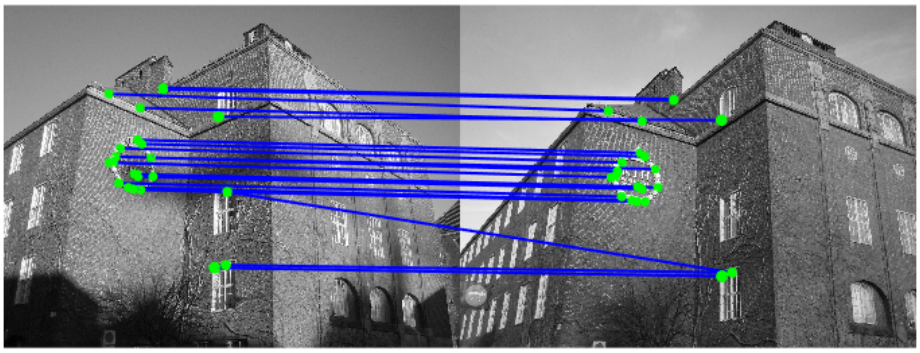

Figure 13: SIFT nearest neighbor distance ratio feature matching (ratio=0.9).

3.2.5 e

In the last section we implemented the SURF with the "nearest neighbor distance ratio" algorithm. With a threshold set to 0.65 we observe that 2 matches are wrong over 8, so the performances here are lower compared to SIFT. Anyway, the "nearest neighbor distance ratio" algorithm gives good result for matching.

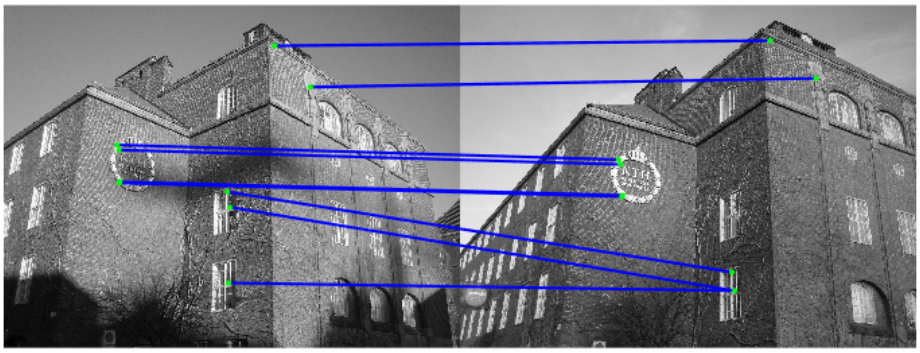

Figure 14: SURF nearest neighbor distance ratio feature matching (ratio=0.65).

4 Conclusions

Overall the sections of the project, our outcomes were better under the SIFT algorithm. On the rotation analysis we observed how SURF performs exactly the same every 90 degrees. On the other hand, in the middle points it performs worst than SIFT. On the scaling analysis, we got unusual behavior, and the matching does not reflect what we saw during lectures. We think that this can be due to wrong assumptions given the up-scaling function. After deep checking of our code we could not find any issue that could have led to such an outcome.

In the second section we also had this distinction between SIFT and SURF with better results for the first algorithm. This outcome can be explained by the fact that SIFT compares 128 values for a single feature, while SURF just 64. With more descriptor values, the matching will be more precise. Anyway these values are not of the same type for the 2 algorithms, so is not always true that more values give better results. On the other way, SURF performs faster and is more adaptable for real-time and low computational capabilities. SIFT is also known for intensively using the RAM to allocate the data needed, resulting in probable implementation constraints.

Appendix

Who Did What

The work was divided equally based on the previous knowledge of the components and the work was organized and developed in parallel.

References

[1] Herbert Bay et al. "Speeded-Up Robust Features (SURF)". In: *Computer Vision and Image Understanding* 110.3 (2008). Similarity Matching in Computer Vision and Multimedia, pp. 346–359. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2007.09.014. URL: https://www.sciencedirect.com/science/article/pii/S1077314207001555.

[2] Bay H., Tuytelaars T., and Van Gool L. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. 2006. DOI: 10.1007/11744023_32.

[3] *ICT business*. https://www.vlfeat.org/install-matlab.html. ultimo accesso 16/09/2021.

[4] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60 (2004), pp. 91–110.