COMPUTER VISION LAB ASSIGNMENT 2

March 5, 2021

Sjoerd Bruin (s2563304) & Jasper van Thuijl (s2503204)

University of Groningen

1. INTRODUCTION

The Scale-Invariant Feature Transform (SIFT) algorithm that was presented by Lowe (1999), can be used in object recognition applications. SIFT transforms an image into a collection of feature vectors that are invariant to a large number of common image operations such as translation, rotation and scaling. Furthermore, the algorithm is stated to be partially invariant to 3D projection, affine transformations and illumination changes. This makes the algorithm quite useful for object recognition applications when the orientation and size of the object to be found does not exactly agree with the subject image. In this report several experiments are discussed in order to establish the effectiveness of the SIFT method in extracting features from images, and how well it performs in matching applications.

2. EXERCISE 1

Using the SIFT method on the provided file ‘scene.pgm‘ results in 1021 keypoints being found. This is in line with the statement by Lowe (1999), that for each image in the order of 1000 SIFT keys are being generated. The positions of these keypoints are shown in Figure 1. From the figure it becomes clear that keypoints are more closely distributed around parts of the image that contain a large number of details. The parts of the image with few details such as the background in the scene contain fewer keypoints. This is in line with the expectations, since SIFT uses the positions maxima or minima of a difference-of-Gaussian function in scale space for generating keypoints. The difference-of-gaussian works by first gaussian filtering the input image. Then a incremental pyramid is created. To create the next slice of the pyramid gaussian filtering is applied, after which the difference between the new layer and the previous layer is calculated. Subsequently 1.5 resampling of the previous layer image is performed. To generate keypoints each pixel is compared with it’s neighbouring pixels; if a maxima or minima exists, the closest pixel location (after correcting for the resampling) is checked again in the next layer of the pyramid. The background consists of relatively large equal parts, resulting in relatively few maxima and minima.

We have matched scene.pgm to the file book.pgm, which

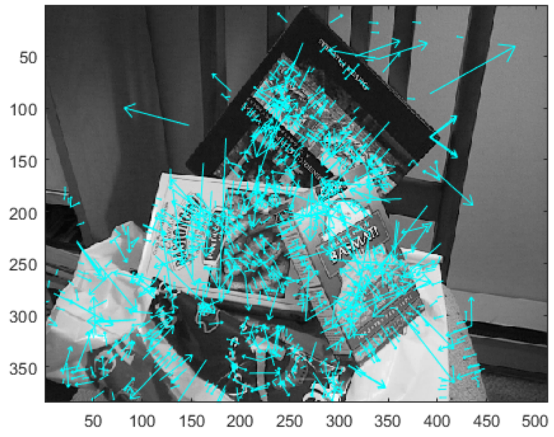

Fig. 1: Keypoints in ‘scene.pgm‘.

is an image of the black book that is visible in the scene. SIFT quickly in computes the match and results in 98 matches. This is equal to about 10% (98/1021) of the scene keypoints, and 11% (98/882) keypoints in the book file. In the output image of the program, which is shown in Figure 4 the found matches are visualized. Except for two unrelated features, the matches seem to be accurate, and closely match the key positions found original book image. The two mismatches can be logically explained, as a similar pattern with similar color is recognized.

Next we analyzed the matching performance of the scene with the picture of the basmati rice box. This results in 1021 keypoints being found and matches 3% (34/1021) of keypoints in the scene image, with 6% (34/579) of keypoints in the basmati image. There resulting output image is shown in Figure 2. Similar to the matching experiment with the book image, the matched keypoints seem to be accurate, except for one unrelated feature. Also in this case, the mismatch can be easily explained due to a similar structure and color.

Next we analyzed a picture of an outdoor scene with a set of five detail cutouts originating from this picture. Both a high quality and low quality version of the outdoor scene image were compared in order to observe the impact of image resolution. In the low-quality comparisons, three of the cutouts resulted in only correctly matched features, while 2 pictures had one or two mismatching features (detail1 and detail4). However with < 37 matching keypoints per image there are only relatively few matching keypoints in the set of images. In the high-quality experiment, a much larger number of matching keypoints are found. However, this does not necessarily mean that these matches are correct. In all of the resulting images mismatches do now occur. Some of the pictures which had a 100% correct match now have a high number of mismatches. Increasing the resolution thus seems not necessarily lead to a better matches: even though the number of matches is higher, the accuracy seems lower.

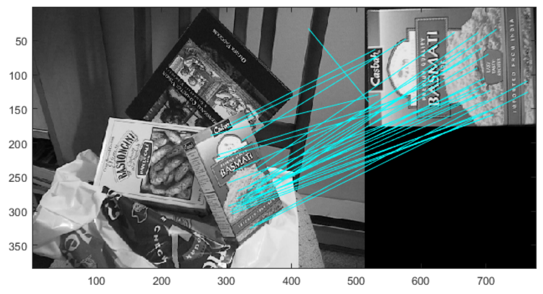

Fig. 2: Matches for the scene image with the photo of the basmati box.

3. EXERCISE 2

In this experiment we tried to match the detail cutout images from the outdoor scene to the indoor scene and the other way around (matching the book and basmati image to the outdoor scene). Since each of these reference images do not occur in the respective subject image, this experiment effectively tests the number of false positive matches that occur in these cases. In the case of the indoor scene, only two matches found for one image, one match in another image and zero matches for the other three cases. For the outdoor case only a single match was observed per image with the book and basmati box images. In the case of the high resolution outdoor image, six matches were observed for the book image, while three matches were observed for the basmati box image which can be observed in Figure 3). Together with the previous results, this indicates that for automated applications one could potentially set a threshold of minimum matches, since the number of observed true positive matches in Exercise 1 were all higher. However, in determining this threshold, the subject image's resolution and quality should be kept in mind, as for high resolution images a larger number of (false) matches can be expected. This can be explained by the fact that high resolution/high quality images most likely also contain more details compared to their low resolutions/low quality variants, which results in more keypoints being generated. The chance that a feature vector with similar value occurs and has a match with one of the reference feature vectors in the nearest-neighbours indexing of SIFT is increased by having a larger number of keypoints. Using just the percentages relating to the number of matches compared to the number of keypoints reference image in the threshold comparison is not necessarily meaningful since the number of matches is also largely specific to the input image. Implementing the threshold to robustly make a match might thus be difficult as it is largely dependent on the input image and reference images being used.

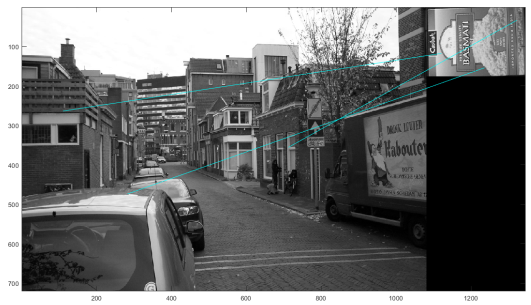

Fig. 3: Mismatches occurring in the high-quality outdoor scene image with the reference image of the basmati box.

4. EXERCISE 3

We have been provided with a set of rotated images of the book.pgm file, with the angles given by:

$$\alpha \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 180, 270\}.$$

Our goal is to match these rotated images to the scene.pgm indoor scene in the same way we have done in previous images, in order to assess the impact that rotation of the reference image has on the recognition capability. We can simply change the definition we have used previously, namely by changing matches = match('scene.pgm', 'book.pgm'), into matches = match('scene.pgm', 'bookr$\{\alpha\}$.pgm), where α is the angle we are currently looking at. The results are shown in table 1.

Rotation	**Matches**
0	98
10	102
20	95
30	107
40	101
50	100
60	102
70	102
80	107
90	103
180	98
270	100

Table 1: Number of matches versus the angle of rotation of the book.pgm image.

From table 1, it becomes clear that the number of matches stays approximately constant under rotation. As such, the ability to match an image to a scene appears to be invariant under the rotation of the reference image. That is, the SIFT procedure is rotation-invariant. This is not surprising: the SIFT algorithm constructs features based on the gradient orientation. What it does is it first determines the cardinal orientation (the orientation with the largest weight associated with it). After that, it determines the distribution of gradients relative to that cardinal (largest-magnitude) orientation, and stores them in a histogram-like feature descriptor. This way, the gradient direction are always relative to the major orientation. This means that we can find the feature desciptor regardless of its primary orientation, since the feature descriptor is described relative to the primary orientation. All we need to do is find the primary orientation of a point, and align that orientation with the cardinal orientation of a feature descriptor. Then, we can match up the gradient orientations around that cardinal orientation with the gradient orientations in the image in order to find a match (or mismatch).

The reference image's performance is shown in figure 4. We see that a lot of matches happen, and the mapping seems to be correct.

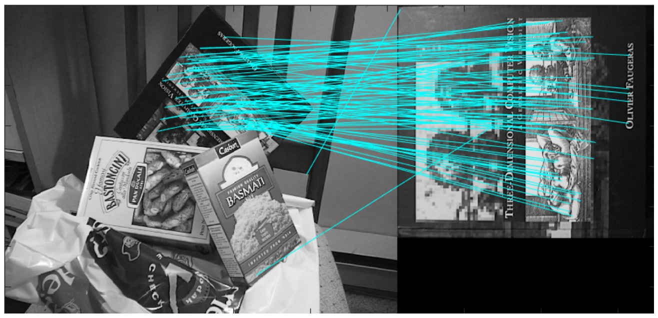

Fig. 4: Matches for the reference image to the scene.

When we rotate the image by 80 degrees, as shown in figure 5, we see that the matches are still correct, and we still have a lot of them. This matches the results we presented in table 1.

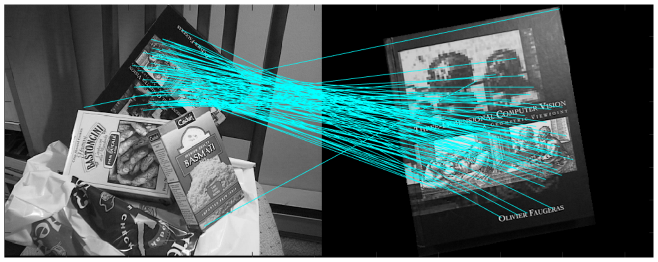

Fig. 5: Matches for the rotated image ($\alpha = 80$) to the scene.

To summarise, the above discussion explains why the SIFT procedure is rotation-invariant. The result of this is clear from table 1: the accuracy of the matching algorithm is independent of the angle by which we rotate the reference image.

5. EXERCISE 4

In this exercise, we perform a similar task as we did in exercise 3. This time, however, we are provided with sheared, rather than rotated, images, with a shear s of the given pixel sizes:

$$s \in \{25, 50, 100, 200, 300, 400\}.$$

We again try to match these sheared images to the original scene, using the original, non-sheared image as a reference against which we judge the performance of the SIFT algorithm on sheared reference images. This operation is quite easy: we simply have to call the command matches = match('scene.pgm', 'books$\{s\}$.pgm'), where s is the number of pixels by which the reference image is sheared. The results of this are shown in table 2.

Shear	**Matches**
0	98
25	100
50	91
100	42
200	3
300	0
400	0

Table 2: Number of matches versus the number of pixels used as the shear distance of the book.pgm image.

The results in table 2 are unambiguous: as the shear size increases, the number of matches decreases. For smaller shear distances, the impact is not yet noticeable (for example for 25 pixels). However, for larger shear distances, we go towards zero matches in total. We saw in exercise 3 that the

SIFT algorithm is rotation-invariant, and we explained it by noting that the gradient histogram around the principal orientation remained intact. When shearing, however, this does not happen: because the structure of the reference image is displaced non-uniformly (due to the shear operation), the gradient vectors are changed as well. More specifically, because the relative location of grey values is changed by the shear operation, it follows that in general the gradient orientation are non-uniformly distorted as well to accommodate for this. As a result, the feature descriptor extracted from the sheared reference image is no longer similar to the one extracted from the unchanged reference image, and as such finding examples of the normal reference image in an image becomes increasingly more difficult. For small shears, we see that the impact is not too great, since the gradient will have undergone only minor changes (because the shear is not that large). However, for larger shears, the gradients diverge from the original gradients too much to allow for recognition of the original image from this sheared reference image and the associated ('sheared') feature descriptors.

The results for difference shears are shown in figures 6-8. In figure 6, we indeed see that most of the matches still are accurate compared to the reference in figure 4, and there are roughly the same number of them compared to the reference image. As such, this small shear does not yet harm the performance of the SIFT algorithm.

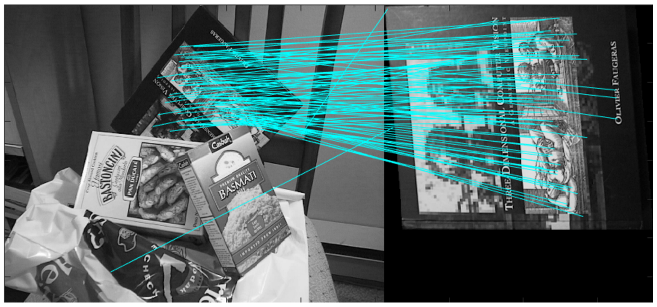

Fig. 6: Matches of reference image with $s = 25$.

At a shear of $s = 100$ as shown in figure 7, we see that the number of matches has decreased quite a bit, although their positions still appear intuitive. It seems that the performance is gradually declining.

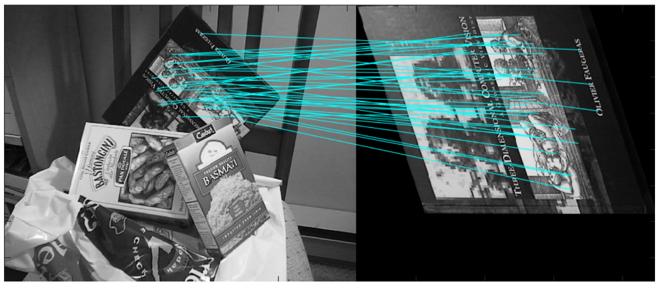

Fig. 7: Matches of reference image with $s = 100$.

Finally, at a shear of $s = 400$ as shown in figure 8, there are no matches whatsoever. By this point, the SIFT algorithm has become completely useless for matching the reference image to examples in the scene. This really highlights that the SIFT algorithm does not work well under shearing of the reference image.

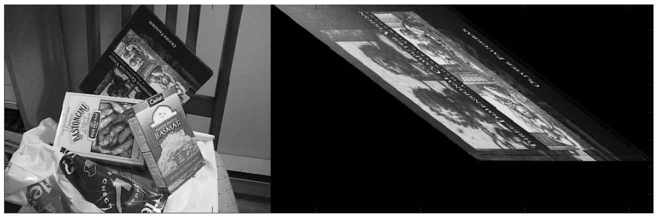

Fig. 8: Matches of reference image with $s = 400$.

The same problem would appear if we sheared the scene image, instead of the reference image, because the gradients in the image would be distorted, and as such the feature descriptors could still not be matched to the examples in the scene. In summary, we found that the SIFT algorithm is not shear-invariant, and that the performance becomes really poor when a significant shear is applied to reference image.

6. CONCLUSIONS

1. The SIFT method is very accurate and fast in object matching applications, even though the objects size, position and orientation might be different. Rotation, translation and scaling of the reference image have almost have no effect on the matching performance. Heavy shearing does however harm the performance of SIFT.

2. In matching applications that use high resolution subject images, a larger number of (false positive) matches should be expected compared to when using the low resolution versions of these images.

3. A threshold could potentially be set in order to differentiate between a true match or a false match; however this is quite difficult as the subject image and reference images resolution & quality can lead to large differences in the resulting number of matches.

References

Lowe, D. (1999). Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1150–1157 vol.2. https://doi.org/10.1109/ICCV.1999.790410