Lab. 4 - Problem-Based Learning: Applications of Invariant Features

Songyou Peng, Èric Pairet
{psy920710, ericpairet}@gmail.com

April 15, 2016

1 Introduction

One of the most well-known algorithms in computer vision is SIFT (Scale-Invariant Feature Transform) [1], which was proposed by David Lowe in 1999. In short, it is a tool which detects the most relevant parts of an image, called keypoints, and describe them through features. It is said that the resulting descriptor is invariant to many factors, such as rotations and scales, among others. Actually, this is the fact which converts SIFT to such a powerful algorithm.

The main goal of this laboratory session is to verify the invariance of the SIFT descriptor and the overall accuracy of the algorithm. Therefore, this report has been structured as follows. In Section 2 it is explained how an appropriate dataset of images has been generated to properly check the performance of the algorithm, which is reported in Section 3. Then, in Section 4, a modification of the algorithm is proposed and its performance contrasted with the one of the original algorithm. Finally, some conclusions are stated in Section 5.

2 Generating the testing dataset

In order to properly check an algorithm, it is indispensable to analyse it with inputs of different characteristics. Therefore, to test the performance of the SIFT algorithm, it is essential to have several images of the same scenario but taken under different conditions. Such a dataset has been created exclusively for this laboratory session, and all related to its creation from the given underwater image is explained next.

The 172 images dataset delivered together with this report has been obtained by applying some transformations and additions of Gaussian noise to the highlighted part of the image shown in Fig. 1. This specific patch of the original image has been chosen because it has both rocky and homogeneous zones, which will provide different amounts and kinds of descriptors.

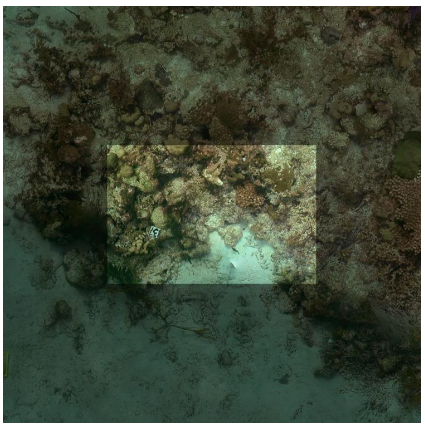


Figure 1: Considered image to generate all the database.

When applying some transformations to the highlighted image it might be needed information from the shadowed part to fully represent the camera change of view. Therefore, the shadowed image has also been considered, if

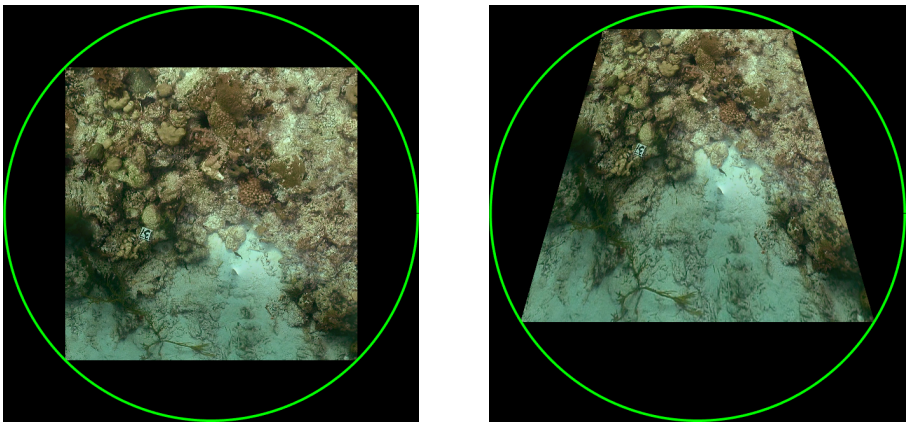
needed, when performing the projecting, the scaling and the rotating transformation, which will be respectively explained in Section 2.1, in Section 2.2 and in Seciton 2.3. Moreover, since an acquired image usually has some induced noise, four differently characterised Gaussian noise distributions have been added to the previously computed images; all with $\mu = 0$ but with $\sigma = 0$, $\sigma = 3$, $\sigma = 6$ and $\sigma = 18$.

All these transformations and additions of Gaussian noise are done in the *main_dataset.m* and *dataset_generator.m* files. These Matlab files not only compute an image, but also the homography which represents such transformation. Another script, which has been delivered with the name *homography_tester.m*, let the user visually check the correctness of the homographies.

2.1 Projective transformation

The images included in the first set simulate different angle viewpoint of the camera. Specifically, we have assumed that the original picture was acquired with a camera pointing down orthogonally to the seabed. This fact means that the camera is placed above the center of the image so the distance from the camera to any of the image corners is the same, as represented in Fig. 2 (a).

If the camera tilts but keeps the same position, the obtained view would be the one shown in Fig. 2 (b), from which a patch of 500×750 pixels can be extracted. Taking into account that the four corners of the considered scenario will always lie on a circle with a known radius, their coordinates can be geometrically computed and therefore, the LMS (Least Mean Square) method can be used to get the transformation matrix. For the sake of simplification, the built-in Matlab function *fitgeotrans* has been used.



(a) Orthogonal view (b) Projective view

Figure 2: Comparison of the camera view.

2.2 Scale transformation

The second set of images emulates the scaling factor when acquiring images at different distances from the seabed. Specifically, the selected patch has been zoomed by setting the boundaries of the image closer to its center and then, resizing it to obtain a patch of 500×750 pixels. This procedure has been performed with scaling factors going from 110 % to 150 % in increments of 5 %, obtaining a total of 9 new images. Moreover, the correspondent transformation have been computed according to Eq. 1, which by specifying the scaling factor s and the center of the resulting image (x_0, y_0) with respect to the image frame, determines the transformation from the fixed image f to the moving one m.

$$ {}^{m}H_{f} = \begin{bmatrix} s & 0 & x_0 - s * x_0 \\ 0 & s & y_0 - s * y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1} $$

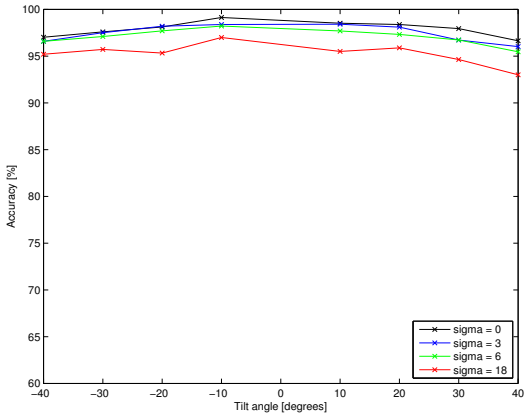
2.3 Rotation transformation

The third set of images consists in different rotational points of view of the same scenario. Specifically, the whole image shown in Fig. 1 was rotated using the Matlab function *imrotate()* with 18 different angles equally distributed between -45 and 45 degrees. After rotating, a patch of 500×750 was cropped from the center of such image. All this procedure has been mathematically represented with the homography shown in Eq. 2, which by specifying the rotated angle θ and the center of the resulting image (x_0, y_0) with respect to the image frame, determines the transformation from the fixed image f to the moving one m.

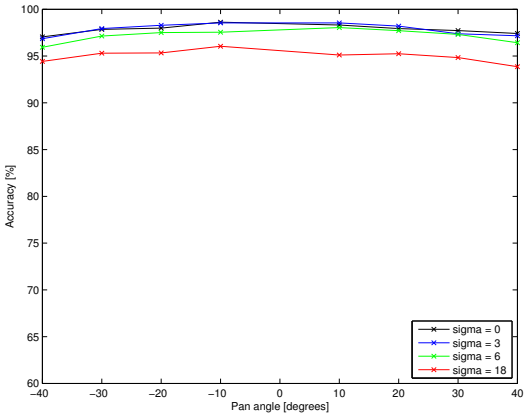
$$ {}^{m}H_f = \begin{bmatrix} cos\ \theta & sin\ \theta & (1 - cos\ \theta) * x_0 - sin\ \theta * y_0 \\ -sin\ \theta & cos\ \theta & sin\ \theta * x_0 + (1 - cos\ \theta) * y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2} $$

3 Testing the performance of SIFT descriptor

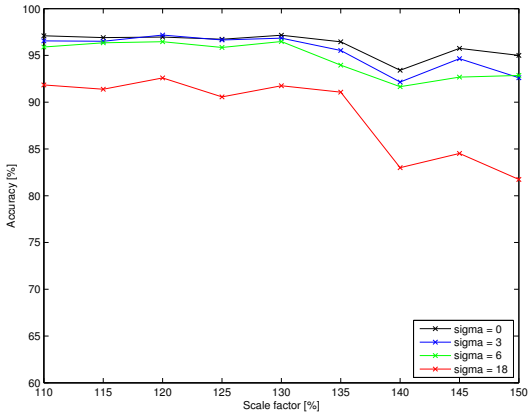
In order to test the performance of the SIFT descriptor, we computed its accuracy under different levels of noise and scenarios, e.g. projection, scaling, and rotation. Specifically, the accuracy is the number of correct matches performed by SIFT out of the number of total matches. For computing all related with SIFT, it has been used the code available in the VLFeat library [2], while for deciding if a match was either correct or not, the homographies computed in Section 2 were used; if the corresponding pair of a keypoint through SIFT and the homographies were at a distance of at most 2 pixels, such matching was considered correct.



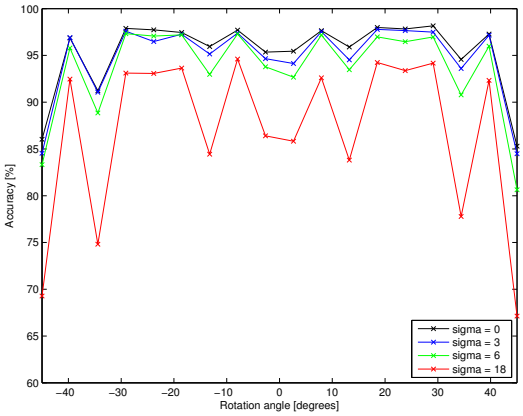
(a) Accuracy when dealing with different tilts.



(b) Accuracy when dealing with different pans.



(c) Accuracy when dealing with different scales.



(d) Accuracy when dealing with different rotations.

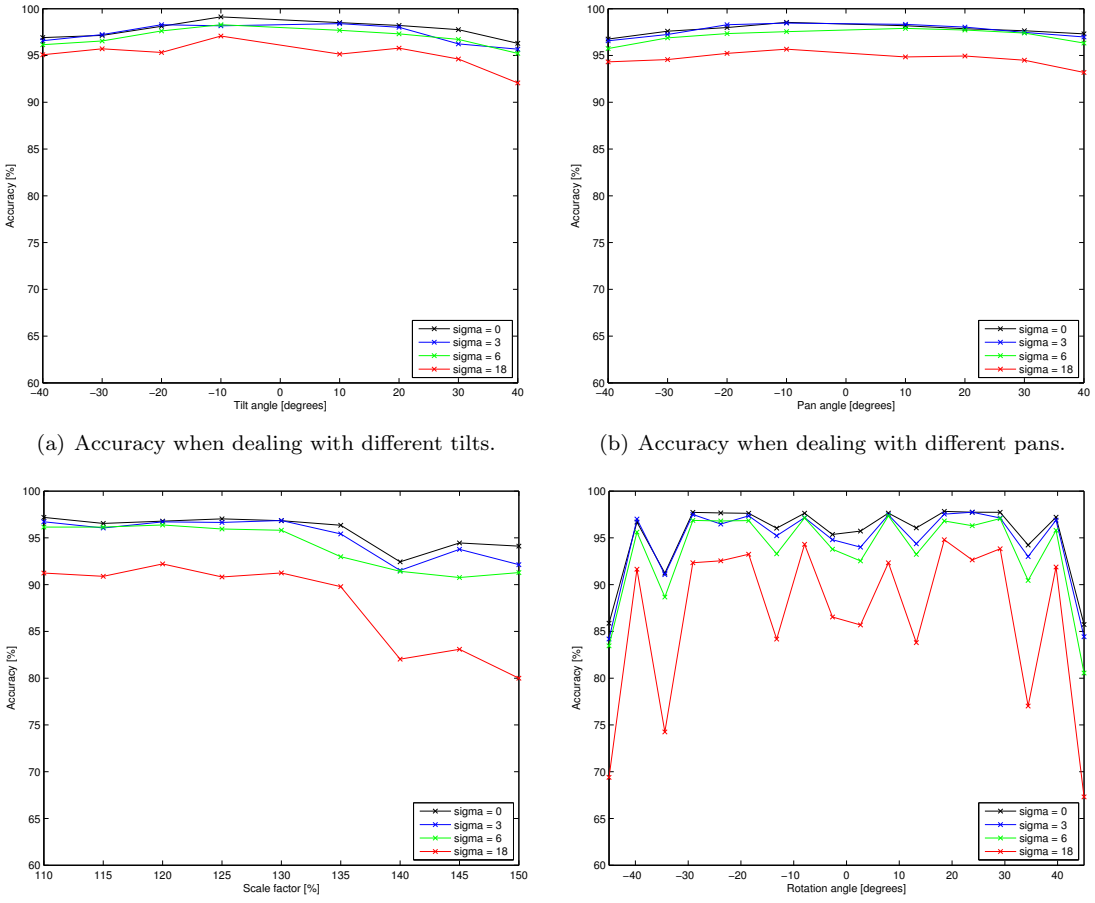
Figure 3: SIFT descriptor accuracy under different conditions.

All the data obtained when looking for correspondent keypoints between the original image and any of the 172 images of the previously computed dataset, has been plotted in Fig. 3. Specifically, each plot shows the dynamic of the SIFT descriptor accuracy when dealing with different levels of a specific variation. Moreover, in different colours, such dynamic is represented when facing noisy images with different amounts of Gaussian noise. In general terms, it can be seen that when increasing the level of noise in any of the four scenarios, the accuracy of the SIFT descriptor diminishes.

Regarding the accuracy when changing the projective viewpoint, the matching rate should decrease when increasing the viewpoint angles. Actually, our result decreases indeed, but not much; since this transformation has been properly modelled, its effects can be compared to a translation and a tiny rotation, to which SIFT is robust. In relation to the effect of scaling, it is vividly demonstrated the theoretical invariance of SIFT to this effect, which is thanks to the DoG (Difference of Gaussians) performed in the detection step. Finally, it has been corroborated that, in average, SIFT is invariant to rotations too. However, their dynamic presents some discontinuities, which are attributed to the change in the number of good descriptors; in some angles, an important part of the image is a homogeneous underwater zone.

4 Implementing a modified SIFT descriptor

In this section, the accuracy of the SIFT descriptor when considering a window of different characteristics from the one in the original algorithm has been studied. Specifically, as *Team A*, we were asked to compute the SIFT descriptor with a window of 12×12 sub-windowed in 3×3 patches.



(a) Accuracy when dealing with different tilts.

(b) Accuracy when dealing with different pans.

(c) Accuracy when dealing with different scales.

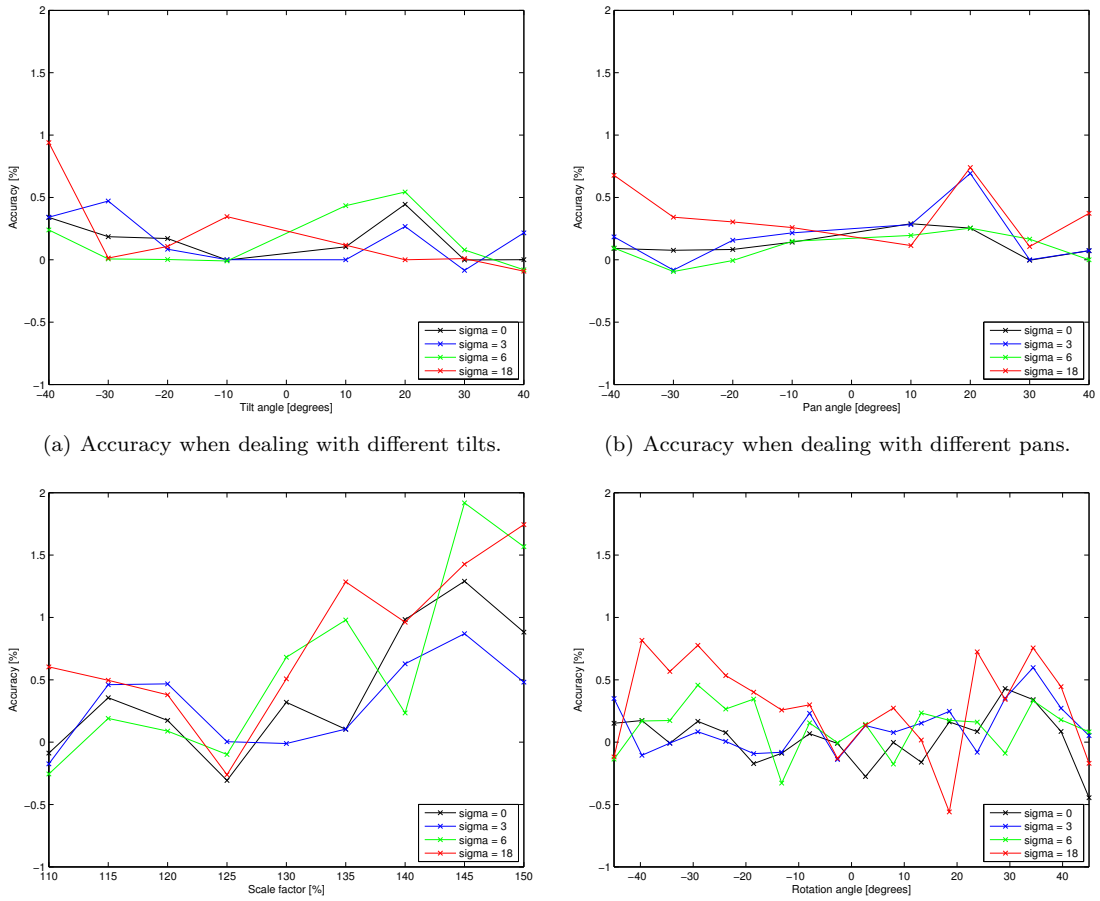
(d) Accuracy when dealing with different rotations.

Figure 4: SIFT descriptor accuracy under different conditions.

Modifying the original window of 16×16 for the one previously described has been possible thanks to the parameter *windowsize* of the *vl_sift()* function, from the VLFeat library. Specifically, this parameter is called the radius of the sub-windows, and by default is set to 2; changing it to 1.5 produces a window with the desired size. At this point has to be said that many approaches and codes have been considered to perform this modification since not all open source codes are compiling in all versions of Matlab.

Even though this modification, the length of the SIFT descriptor will still be of size 128. However, less information about the neighbourhood of the keypoint is encoded in the descriptor and the accuracy ought to slightly decline. In order to proof this theoretical statement, the same experiments performed in Section 3 were repeated, but considering the windows of 12×12.

A visual comparison of Fig. 3 and Fig. 4 does not give much information about the change of accuracy when considering a smaller window. Therefore, in order to get meaningful information to compare the accuracy under these two window parametrisation, the difference between the previous results has been computed and plotted in Fig. 5. Specifically, the accuracy of the modified SIFT has been subtracted to the accuracy of the original implementation of SIFT.



(a) Accuracy when dealing with different tilts.

(b) Accuracy when dealing with different pans.

(c) Accuracy when dealing with different scales.

(d) Accuracy when dealing with different rotations.

Figure 5: Modified SIFT descriptor accuracy under different conditions.

The results shown in Fig. 5 demonstrates that a higher accuracy is generally achieved when considering a window size of 16×16 instead of one of 12×12. However, even though this fact should theoretically happen in any case, the obtained results reveal that under some situations the modified version performs slightly better. Moreover, we also test the accuracy of the SIFT descriptor with a 8×8 window divided by four 2×2 sub-windows; in such experiment, the reduction of accuracy is way more aggressive and significant.

5 Final remarks

In this report, an extensive evaluation of the SIFT descriptor performance when dealing with different levels of Gaussian noise and transformations has been carried out. For this purpose, it has been indispensable to build a dataset of images representative to all the invariances that had to be tested, e.g. projection, scale and rotation invariance. The correspondent source code has been developed in Matlab, which can be revised in the *main_dataset.m* and *dataset_generator.m* files. Moreover, an extra script called *homograpy_tester.m* has been developed to check the correctness of the homographies.

The other key component of this laboratory session is the SIFT algorithm. In this case, the implementation available in the VLFeat library has been integrated into the *main_sift.m* file to perform all the required tests. As mentioned in [3], the large majority of keypoints correspond nearly exactly to the ones obtained with the original implementation. Specifically, the center of a descriptor computed with both codes would match with a precision of 0.01 pixels the 89 % of the times. Regarding the differences on the descriptor itself, it is said that over the 98 % of the descriptors computed through the VLFeat implementation have less than the 20 % of the average descriptor distance.

These two components have allowed to carry out all required experiments. The first set of trials has corroborated the invariance of SIFT to scales, rotations, and different viewpoints. Even though this statement is almost true in all cases, the performance of the SIFT descriptor when dealing with high levels of Gaussian noise and large scaling factors or rotations can be decreased up to a 20 %. However, it has to be said that under normal conditions it is hard to face a Gaussian noise as high as the biggest one evaluated in this report.

Regarding the accuracy of the SIFT descriptor when considering a window of 12×12 instead of the original one of 16×16, it has been stated that, as expected, it decreases. However, in some trials, it has been seen that the accuracy of the modification is slightly better. In order to corroborate this fact, further trials should be done with different images to analyse the average accuracy of these two descriptors.

To sum up, in this laboratory session, different image transformations have been considered to obtain a meaningful dataset to test the SIFT descriptor invariance. Then an interesting insight of the SIFT descriptor accuracy has been reported under different conditions. Finally, the optimality of the window size has been partially tested by analysing the accuracy of the same descriptor but when changing its window size.

References

[1] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[2] VLFeat open source library. `http://www.vlfeat.org/`. Last access: 05-05-2016.

[3] SIFT detector and descriptor. `http://www.vlfeat.org/overview/sift.html`. Last access: 10-05-2016.