University of Girona

Visual Perception

Lab 4

Applications of Invariant Features

Author:
Di MENG
Darja Stoeva

Supervisor:
Rafael Garcia
Ricard Prados

May 29, 2017

1 Introduction

As we know, SIFT is a famous algorithm in computer vision developed by David G. Lowe to detect local features in images. The descriptors computed by this algorithm are scale invariant, orientation invariant and also partially illumination and 3D views invariant. For better understanding the functionality of SIFT algorithm, the objective of this lab assignment is to test the algorithm and analyze the performance.

The report is divided into three sections. The dataset with different scales, rotations and views is built for the first section, followed by testing the code provided by David G. Lowe to observe the performance in second section. The last section is to implement our own SIFT algorithm and analyze the difference.

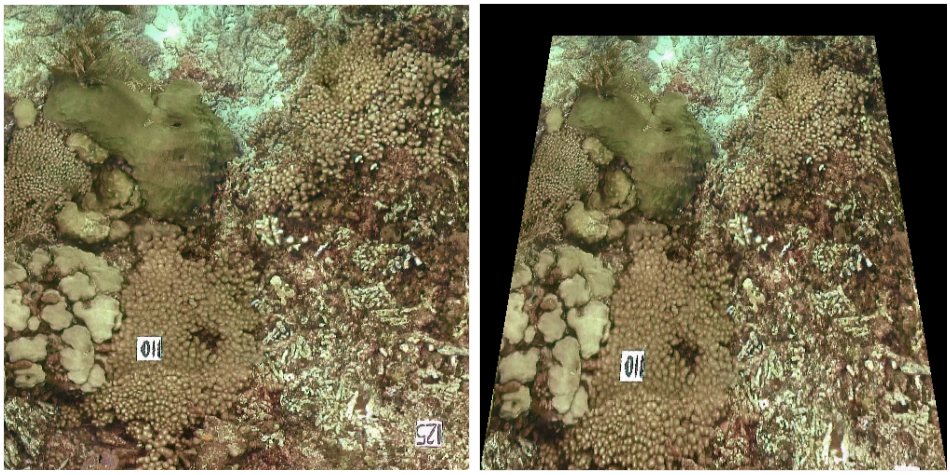
2 Dataset generating

Since the descriptors computed by SIFT are scale invariant, orientation invariant and 3D view invariant, dataset with those different transformations has to be generated for testing the SIFT properties. Here, we generate three sequences which used projective transformation, scaling and rotation transformation. Each sequence is a structure with homographies for each image.

2.1 Projective transformation

Projective transformation can describe a camera motion which is tilting the camera up and down, left and right. What we do basically is taking a square image form the original seabed image, keeping the center position and indent one edge of the square. Here, we use four pairs of correspondence coordinates which are the four corners of original square and projective one. By using Least Square Method, we obtain the homograpgy matrix which transformed the original square into projective view. For keeping the center position, the inverse homography is applied to the center of the projective image. Then the center obtained which is not lying in the center of the original square should be corrected. Thus, a translation matrix taking the distance of two centers is applied to the homography transformation.

After transforming the image properly, the central part of the transformed image is cropped which is the generated image. The correspondence homography can be obtained by taking the four pairs of corners of the cropped images. In addition, for each transformed image, four levels of noise are added into it.



(a) Original square (b) Projective Image

Figure 1: Projective transformation

2.2 Zoom

For generating different scale images, nine scale factors are used to build the dataset. With different scaling factor, we have closer view of the seabed. The center position is kept the same. Thus, the homography matrix can be obtained as follows. Also, each image is added different levels of noise.

$$H = \begin{pmatrix} s & 0 & x0 - s * x0 \\ 0 & s & y0 - s * y0 \\ 0 & 0 & 1 \end{pmatrix}$$

Where x0 and y0 are the center position of the image, s is the scale factor.

2.3 Rotation

The same as the zooming transformation, the center has to be kept the same position when doing the rotation transformation. In order to do that, we

have to translate the center to the left corner of the image and then translate it back after applying the rotation transformation. So, the homography is the multiplication of three transformations.

$$H = \begin{pmatrix} cos(\theta) & -sin(\theta) & x0 * (1 - cos(\theta)) + y0 * sin(\theta) \\ sin(\theta) & cos(\theta) & -x0 * sin(\theta) + y0 * (1 - cos(\theta)) \\ 0 & 0 & 1 \end{pmatrix}$$

Where θ is the rotated angle and x0, y0 are the center positions.

2.4 Testing the sequence

In order to make sure the homography in the sequence we built provides the correct transformation, we pick up one point in the original image and apply the homography to it to see if it goes to the matching position.

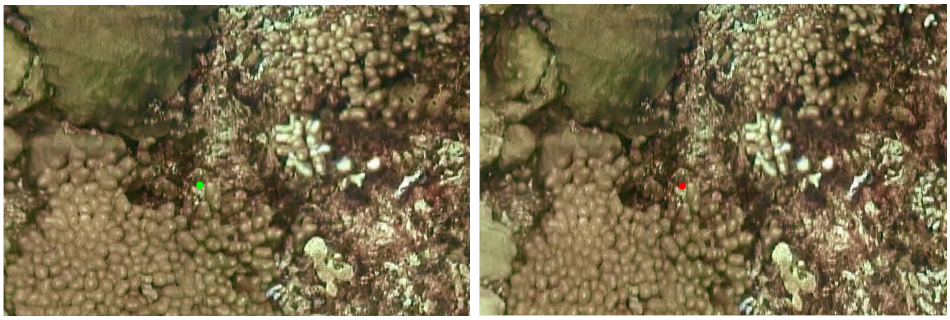


Figure 2: Matching points with projective transformation

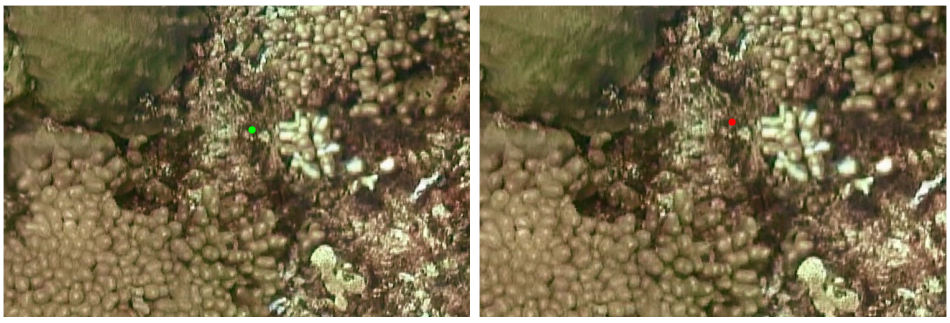


Figure 3: Matching points with scaling transformation

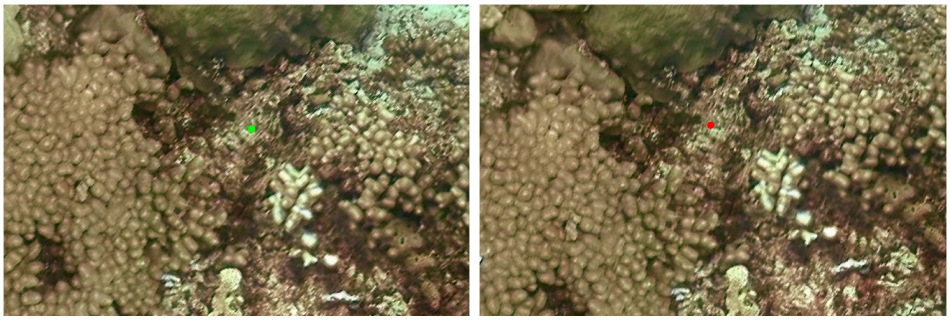


Figure 4: Matching points with rotation transformation

3 SIFT performance

Since we have the dataset which contains different transformations of one original image, we use this dataset to test the performance of SIFT implementation by David G. Lowe.

For each transformation(projective, scaling, rotation), we have one basic image and the rest are the ones with different factor and noise. According to each pair of the original one and the one from the rest, we compute the matching keypoints of two images by SIFT. Among these matching keypoints, we apply homography which gives us the correct corresponding keypoints' locations to check how many keypoints are matched correctly.

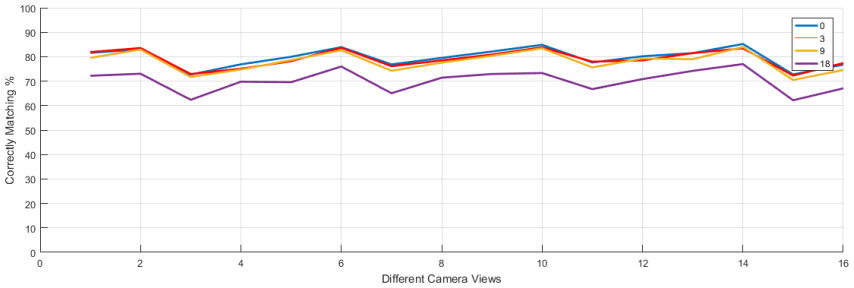


Figure 5: Correctly matched percentage along different camera views.

If the matched points computed by SIFT locates at the position of points computed by hompgraphy with 1 pixel tolerance, we consider the location is correct. From the figures, we note that SIFT gives excellent matching accuracy.

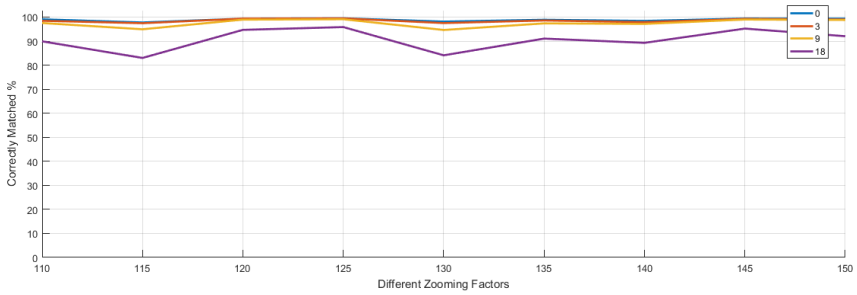


Figure 6: Correctly matched percentage along different scale factors.

4 Implementation of our own SIFT

We chose team A which is to implement two kinds of version of SIFT, one is with standard 16x16 size of descriptors and 4x4 subwindows, the other one is modified version with 12x12 size of descriptors and 3x3 subwindows. In this case, we use vl_feat library which is also working SIFT algorithm but not implemented by the author. The default parameter 'window size' for standard version is 2. It is set to be 1.5 for getting 3x3 suwindows. The comparison and performance are tested.

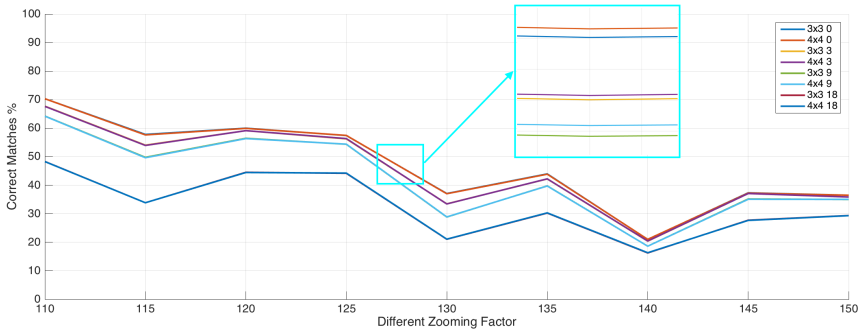


Figure 7: Comparison between standard SIFT and modified version according to zooming factor.

We can see that the performances of two versions are almost the same, the standard version provides slightly better result than the modified one.

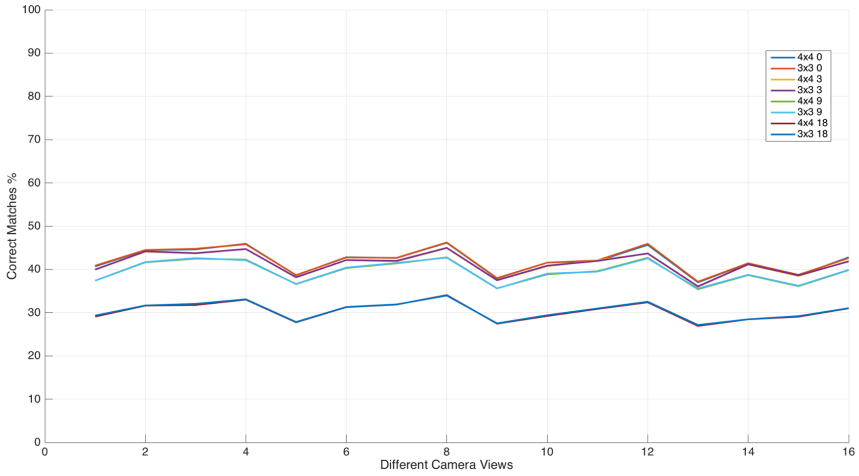


Figure 8: Comparison between standard SIFT and modified version according to different camera views.

5 Conclution

A dataset with different kinds of transformations is built for testing the performance of SIFT algorithm. By observing the correctly matched accuracy, SIFT gives good matching which means it is scale invariant, orientation invariant and 3D view invariant. It provides powerful local descriptors to find and match the key features. What's more, a modified version of SIFT is implemented by ourselves. After comparing their performance, we note that the standard SIFT performs slightly better than the modified one.