# Visual Perception

Lab on Problem-Based Learning:
*Applications of Invariant Features*

Rafael Garcia and Ricard Prados

Related to Lecture 4

| | |
|---|---|
| **Required effort:** | 50 hours |
| **Programming platform:** | Matlab |
| **Version:** | 2.0 |

## 1. Objective

The aim of this lab is to develop competences regarding teamwork and problem solving. By developing the proposed activity you will also become familiar with SIFT: how to extract invariant features, how to describe them, how to match them. Most importantly, this should give you some feeling about the strengths and weaknesses of local feature-based approaches.

It should be noted that before you start this activity, you should read Lowe's paper on SIFT. The paper can be downloaded from here:
http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf
Ref: David G. Lowe, **"Distinctive image features from scale-invariant keypoints,"** *International Journal of Computer Vision,* 60, 2 (2004), pp. 91-110.

This lab requires working in teams of 2-3 students. The labs will be organized on the first lab session by the professor who acts as lab instructor.

Students will have to:

1. Create a testing dataset intended to verify the invariance of the SIFT descriptor.
2. Test Lowe's code with a set of images and produce results about the percentage of correct matches for every pair of images.
3. Implement a modified version of the SIFT descriptor and compare it with the results obtained using the standard SIFT implementation (using Lowe's code).

## 2. Generating the testing dataset

Step 0

In order to evaluate the performance of the SIFT descriptor, three image sequences, belonging to the same synthetic underwater scene, will have to be generated. Each sequence will be composed by:

1. A reference image, which is common for all the sequences.
2. A set of images describing different camera movements around the same region of interest of the scene.

Each image sequence will represent a different type of camera motion. This will allow appropriately evaluating the invariance of the descriptor to these different transformations.

**SEQUENCE 1 (*projective transformation*):** This sequence has to be composed of a set of underwater in which the optical axis of the camera is initially orthogonal to the observed plane, and for every image, the **viewing angle** of the camera **moves away** from the initial position, increasing the tilt angle. The sequence has to contain 16 images, resulting of tilting the camera left and right at four different angles around the scene axes. The homography matrices will be stored on a .mat file which will be called *Sequence1Homographies.mat*, in a structure with the same name of the file.

**SEQUENCE 2 (*zoom*):** This sequence has to be composed of a synthetic set of images in which, for every image, the **distance between the camera and the scene has decreased**. The sequence will contain 9 images, as a result of doing a zoom of the scene from 110% up to a 150% in increments of 5%. The homography matrices will be stored on a .mat file which will be called *Sequence2Homographies.mat*, in a structure with the same name of the file.

**SEQUENCE 3 (*rotation*):** This sequence has to be composed of a set of underwater images in which for every image the camera rotates along its optical axis according to regular increments. The sequence will contain 18 images, resulting of rotating the camera around the center of the scene, ranging from -45 to 45 degrees. The homography matrices will be stored on a .mat file which will be called *Sequence3Homographies.mat*, in a structure with the same name of the file.

In order to generate the three datasets, the image **Image_base.jpg** will be used as a base. It can be downloaded from:

```
server:   senglaret.udg.edu
port:     31422
username: VP2014
password: VP2014
```

From this image, a cut of an interest region of 750 x 500 pixels will be performed, and named **Image_00a.png**. This image will be used as a reference for all the datasets.

As you know, a homography matrix allows describing a planar transformation between two images. Given a planar transformation between images 0 and n, the corresponding homography matrix will be defined as ${}^{n}H_{0}$, that is, a matrix that allows obtaining, from a point represented in the image frame 0, its coordinates on image n. Formally, this can be represented as follows: ${}^{n}p = {}^{n}H_{0} \cdot {}^{0}p_{i}$. The homographies describing the transformations from image 0 to image n will be stored in the structures SequenceXHomographies (where X is the number of the sequence) mentioned above, concretely in a matrix called H. Additionally, every image will have versions (a), (b), (c) and (d) where (a) corresponds to the image without noise, (b) indicates that the image has additive 0-mean Gaussian noise with standard deviation of 3 gray values, (c) means that the image has additive 0-mean Gaussian noise with standard deviation of 6 grayscale values, and (d) corresponds to a noise with a standard deviation of 18 grayscale values. This information will be embedded in the filename. For instance, image 2 without noise would be **Image_01a.png**.

Each sequence will be stored in a different folder named SEQUENCEX, containing the reference image Image_01a.png, the generated dataset images and the structure storing the homography matrices. Figure 1 show an example of code where the homography storage structure is loaded, the reference and a given image sequence are loaded, a given point coordinates on Image 0 are defined, and its location on Image 0 and 4 are displayed.

```
load Sequence2Homographies

Image_00a = imread('Image_00a.png');
Image_04a = imread('Image_04a.png');

p_00 = [316 290 1];
p_04 = Sequence2Homographies(4).H * p_00';

figure; imshow(Image_00a); impixelinfo; hold on;
plot(p_00(1), p_00(2), 'gx');

figure; imshow(Image_04a); impixelinfo; hold on;
plot(p_04(1), p_04(2), 'rx');
```

**Figure 1: Code sample illustrating the file naming criteria and the usage of the homography storage structure.**

## 3. Testing the performance of the SIFT descriptor

Introduction

David Lowe, the researcher who proposed SIFT, has available on his website the binaries of his feature extraction (DoG) and description (SIFT) algorithms, see http://www.cs.ubc.ca/~lowe/keypoints/ ("SIFT demo program"). The package also includes both matlab and C code to visualize the features extracted from an image and to find matches between features found in two different images.

However, in this lab session, we are not only going to ask you to test the performance of Lowe's implementation of SIFT, but also to implement your own descriptor, using the tricks that David Lowe describes in his paper, but adding some variants and testing them against a set of control images.

The control images consist of image pairs in which we know the projective homography that relates every pair of images. This means that when you look for correspondences, you can know if a given pair of correspondences is correct or not by checking if it is close to the provided homography.

Step 1

Download the package siftDemoV4.zip (link "SIFT demo program (Version 4, July 2005)") from Lowe's page.

Step 2

Implement a testing system to generate plots showing the percentage of correct matches for every pair of images of SEQUENCE 1. As an example, Figure 1 shows a plot illustrating the percentage of correct matches for the image pairs corresponding to a change of the viewing angle of 0, 10, 20, 30, 40 and 50 degrees, respectively:

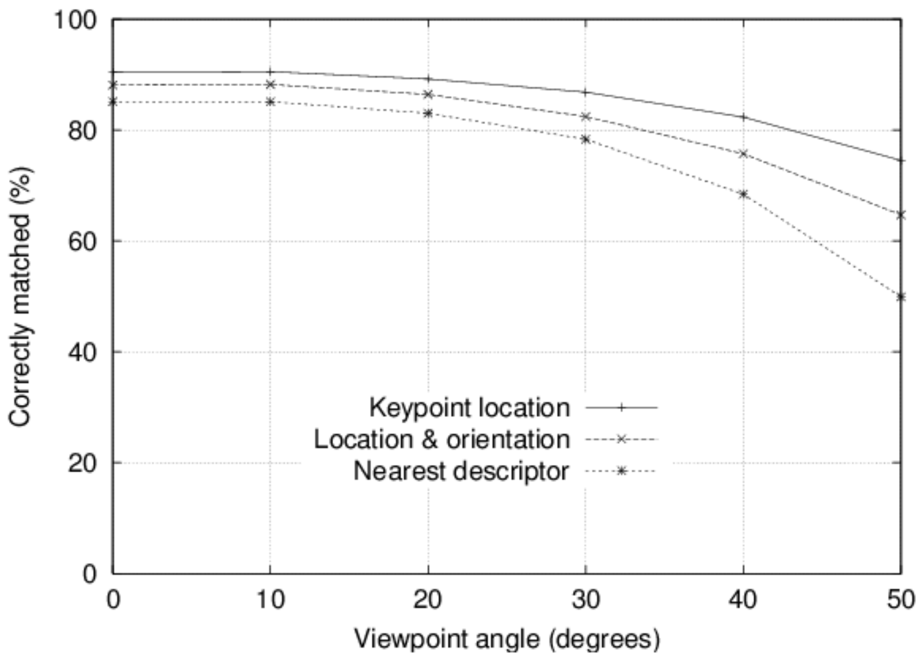

Figure 1: **Sensitivity to changing the viewing angle**

Step 3

Implement a testing system to generate performance plots for every pair of images of SEQUENCES 2 and 3.

## 4. Implementing a modified SIFT descriptor

Step 4

In what follows, you will have to implement two versions of your own SIFT descriptor so that you have different configurations and you can compare its performance (except for Team E, which requires comparing SIFT to other algorithms for solving the correspondence problem).

Hints...

At this stage, you may want to explore what source code is available in Internet (e.g. OpenCV, Vedaldi's implementation of SIFT, etc.).

**Team A**

Implement the standard 16x16 descriptor with 4x4 subwindowing and compare it with your own implementation of a 12x12 descriptor and 3x3 subwindowing. Test them for SEQUENCES 1, 2, and 3 and generate the corresponding tables and plots to visualize the results.

**Team B**

Implement the standard descriptor with scale-space representation and compare it with an implementation where there is no scaling (so that the keypoints are only detected at a single scale). Test them for SEQUENCES 1, 2, and 3 and generate the corresponding tables and plots to visualize the results.

**Team C**

Implement the standard descriptor with main orientation estimation and compare it with an implementation where there is no rotation estimation (so that the descriptor is not shifted according to the main orientation). Test them for SEQUENCES 1, 2, and 3 and generate the corresponding tables and plots to visualize the results.

**Team D**

Compare the performance of SIFT and at least 3 of the following: SURF, GLOH, LESH, BRISK, ORB, FREAK, MOPs and Hugin. Test them for all images of SEQUENCES 1, 2 and 3.

**Team E**

Implement the standard 16x16 descriptor with 4x4 subwindowing and compare it with a 20x20 descriptor and 5x5 subwindowing. Test them for SEQUENCES 1, 2, and 3 and generate the corresponding tables and plots to visualize the results.

## 5. Methodology

You will carry out this activity in groups of 2 (or, exceptionally, 3 students). The groups have been predefined. You need to decide how you are going to organize yourselves to address this activity and how are you going to solve the problem.

A lab instructor will be in charge of every group. The whole group should meet and report weekly to the lab instructor.

## 6. Deliverable

At the end of the activity every group should deliver a report and the code that has been developed. The report will consist in a brief introduction to the problem, and a section corresponding to each of the steps described on this document. The report will be preferably formatted using single column layout, in order to allow an easier placement of the images and figures that you will produce during this exercise.