Laboratory 1:
Image Feature Detection

Computer Vision, Spring 2022

Overview

The goal of this practice is to get familiar with some feature detection algorithms. Particularly, you will study the scale invariant feature transform (SIFT), and then compare more options in state of the art. This practice consists of two parts.

- In the first one you will extract SIFT key-points and descriptors of an image. To do that, you have to understand as a model-based method works.
- In the second one you will compare several feature detection algorithms.

First of all, we must download the codes and supplementary materials from the link *Laboratory 1 Materials* on Aula Global of *Computer Vision*.

FAQs:

- **What do I need to send?** A report with your work, including missing codes, an explanation where you discuss your work, as well as your results and conclusions. Extra codes and analysis are also welcome.
- **How?** A single report is performed for both sessions. Please, do not send partial reports. Once your work is finished, you should upload it to the *Laboratory 1 Submission* on the Aula Global, in a zip file. The submitted zip file has to contain your report and your source code (and all files needed to run it, such as images, other dependencies, etc.). **Be clear and concise.** Your final score is not directly proportional to the number of pages in your report. The work in this session should be done in 2-people groups.
- **When?** Anytime until 14 April 2022 (14:29 UTC+2). After that, nothing will be received.

  Within 24 hours prior to deadline, no comments will be answered.

References

- SIFT slides of the course on *Computer Vision*.
- [1] D. Lowe. Distinctive features from scale-invariant key-points. International Journal of Computer Vision 60(2): 91-110, 2004. You can download it from Aula Global.

1 Introduction

The main aim of this practice is to get familiar with some ideas and concepts on image feature detection.

- You will first read and understand some provided functions that compute the scale space and then you will implement a method to extract key-points by finding extrema in the computed scale space.
- We will use public functions to establish a comparison between some state-of-the-art algorithms.

1.1 SIFT

- Go into the folder "scale_space" and look into the Matlab functions. There are seven Matlab functions:
  - **main.m**: This file contains the main Matlab script. In it, we first define some parameters needed for the scale space computation and then we call three functions which are:
    - **scale_space.m**: This file implements a function that computes the scale space of an image and returns it.
    - **difference_of_gaussians.m**: This file implements a function that takes as input a computed scale space and return the difference of Gaussians.
    - **find_extremas.m**: This file implements a function that takes as input the difference of Gaussians and returns a set of points that are extrema.
  - **gaussian_filter.m**: This file implements a function that takes as input an image and a number representing the standard deviation σ of a Gaussian function and return the image filtered with that Gaussian.
  - **double_image.m**: This file implements a function that takes as input an image and returns the same image scaled by a factor of 2.
  - **half_image.m**: This file implements a function that takes as input an image and returns the same image scaled by a factor of 0.5.
- Tasks [M - Mandatory, O - Optional]:

  M You need to understand what each function does in terms of functionality, input and output. Then you need to implement the missing instructions in the file **find_extremas.m**. The location of the code that needs to be implemented is indicated by the text "MISSING CODE HERE".

  M Try to run your code in several examples, by considering indoor and outdoor scenarios as well as textureless areas. You can acquire your own dataset here, or surf on the Internet. What can you comment about the results? Are there outliers in your estimation? If so, please explain some ideas to remove them.

  O Implement rejection of key-points that have low contrast. See beginning of Section 4 "Accurate Key-point Localization" in the original paper for details [1].

1.2 Feature-detection algorithms

In the previous task you implemented a feature detection algorithm based on SIFT. Now, you will study a public implementation of that detector together with additional ones (such as FAST, SURF, ORB, KAZE, HARRIS, etc.). In the main folder, you can find the functions **detection_cmp.m**. There, you will study functions of the type detectXXXFeatures, where XXX denotes the detection type, SIFT, for instance.

- Tasks [M - Mandatory, O - Optional]:

  M Considering the image *sunflower.jpg*, and for every detectXXXFeatures function, you should analyze the effect of every argument. For example, detectSIFTFeatures could consider a ContrastThreshold, a EdgeThreshold, a NumLayersInOctave and a Sigma parameter. Tune the previous arguments to obtain a good number of features, a good distribution of them, and so on.

  M Provide a table with the results for every feature-detection algorithm, including the computation time in seconds. Could you comment the main characteristics for every solution? For example, for every XXX if blob or corner points are detected, how is the detection in textureless areas, etc. Are the analyzed algorithms scale independent?