Laboratory 1:
Image Feature Detection and Feature Matching: Comparison and Applications

Artificial Vision, Spring 2025

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 *Artificial 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?** Once your work is finished, you should upload it to the *Laboratory 1 Submission* on the Aula Global, in a single 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 23 April 2025 (12:29 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 *Artificial 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.
- [2] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou. LoFTR: Detector-Free Local Feature Matching with Transformers. In Conference on Computer Vision and Pattern Recognition, 2021. You can download it from Aula Global.

1 Introduction

The main aim of this practice is to get familiar with the scale invariant feature transform (SIFT). This practice consists of several parts:

- In part #1, you will use public functions to establish a comparison between some state-of-the-art algorithms.
- In part #2, you will first test David Lowe's code for SIFT (**it runs only on Windows and Linux**) and then compute the homography (or affine map) between two images given a set of matchings computed using SIFT. You will then compare the performance of the algorithm on different types of images, and discuss your results.
- In part #3, you will complete a quantitative and qualitative comparison between state-of-the-art approaches.
- In part #4, you will test a data-driven approach based on Transformers.

1.1 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?

1.2 SIFT+RANSAC

- Go into the folder "Lowe_SIFT" an read the README file.
- Run the code on the sample images present in the same folder (as indicated in the README file). You have to run both scripts **sift.m** and **match.m** and understand their respective input and output.
- Go to the folder "data_set" and look at the images that are inside its sub-folders *bikes*, *boat*, *graf* and *leuven*. Every sub-folder has a set of images and a set of text files. Each one contains the Ground Truth (GT) homography between a couple of images. For example, file "H1to3p" contains homography between images #1 and #3.
- Tasks [M - Mandatory, O - Optional]:
  - M Write a Matlab function that takes as input two images (im1 , im2) and estimates homography between them:

1. computes the SIFT descriptors of each image and the set of matchings between the two images.
2. then calls the function **get_matching_pts** (provided to you) to get the lists in a nice way.
3. then calls the **ransacfithomography** function (provided to you in the "lib" folder). This will apply RANSAC and extract a homography between the two images.
4. returns the estimated homography and an error with respect to GT.

M Write a Matlab script that for every sub-folder does the following:

1. applies the function, implemented previously, to the first image and all other images in a certain sub-folder.
2. for every pair of images applies the estimated homography to the first image using the function *imTrans* (provided to you in the "lib" folder).
3. for every pair of images applies the ground truth homography to the first image using the function *imTrans*.
4. for every pair of images shows both transformed images side by side for comparison. **NOTE**: do not forget to include these side-by-side-comparison figures in your report. One set of figures from any sub-folder of your choice will be enough.
5. for every pair of images measures the error between the computed homographies and the ground truth ones.

M Include all measured errors in your report and discuss their values for different sub-folders.

1.2.1 Real Applications: Mosaics

Now, we apply our knowledge from the last part to solve a real application. Particularly, we will obtain image mosaics from RGB images.

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

M In the folder "Mosaic_SIFT", you have to complete the function **main.m**. Basically, you should provide SIFT matches and compute the homography in the parts indicated by the text "MISSING CODE HERE", by using the functions employed in the current session. In the example, you will obtain an image mosaic composed by two images.

O 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?

O Extending your source code to handle more than two images and obtain a panoramic view.

1.3 Comparison of descriptors

Now we analyze several image detection algorithms for matching. Particularly, we consider the approaches that were seen in part 1 (FAST, SURF, ORB, etc.). To this end, you have to consider the function **match_comparison.m** that it can be found in the main folder.

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

M For the eight descriptors, produce synthetic images considering the input image as a baseline with scale=$\{1.2 : 0.4 : 2\}$ and rotation=$\{-60 : 20 : 60\}$, i.e., (21 pictures in total). Provide the number of matches for every method and case as well as the computation time. According to the obtained results, do you consider the descriptors to be invariant to scale and rotation? Please, note that in some cases the results could not be possible as the number of points is very small or null.

O Provide the same analysis as in the previous point, but this time considering changes on illumination by using real images. It is worth noting that changes on illumination could be considered together with scale and rotation but not mandatory.

1.4 Data-driven Algorithms

Finally, we test a data-driven algorithm where a transformer-based module is considered to learn a feature descriptor.

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

  M Follow LoFTR to run a LoFTR demo on Google Colab. To this end, you need to upload your own image pair, and adjust the image type (indoor/outdoor). For more information, follow carefully the information in the original code.

  M Select up to three image pairs (horizontal pictures with width>height should be considered) where you consider: 1) global and well-textured scenes, texture-less areas, and pictures with motion blur. How many matches can you obtain for every case? Is the feature distribution correct? Please, comment on the results and provide a qualitative evaluation of the results.

  O Comment the main parts of the algorithm to learn the feature descriptor.