CS-E4850 Computer Vision
Exercise Round 4

The following instructions are for the Matlab version. The instructions for the Python version are in github. Matlab is available on Aalto computers and also for students' own computers via `https://download.aalto.fi`.

The problems should be solved before the exercise session and solutions returned via the MyCourses page. Upload TWO files: (1) a PDF file illustrating your results with images and a few lines of text, and (2) a zip file which contains your Matlab source code files.

Get the example `m`-files by downloading `Exercise04.zip` from the MyCourses page.

Exercise 1. Matching Harris corner points.
Run the example file `HarrisMatching.m` and see the instructions in the comments of the source code. The example detects Harris corners from two images of the same scene, extracts image patches of size 15×15 pixels around each corner point and matches mutually nearest neighbors using the sum of squared differences (SSD) similarity measure. The SSD measure for two image patches, f and g, is defined as follows

$$SSD(f,g) = \sum_{k,l} (g(k,l) - f(k,l))^2 \tag{1}$$

so that the larger the SSD value the more dissimilar the patches are.

Do the task (a) below and answer questions (b) and (c):

a) Implement the matching of mutually nearest neighbors using normalized cross-correlation (NCC) as the similarity measure instead of SSD. NCC is defined on the slide 97 of Lecture 2. For two image patches of similar size it can be written as follows:

$$NCC(f,g) = \frac{\sum_{k,l}(g(k,l)-\bar{g})(f(k,l)-\bar{f})}{\sqrt{\sum_{k,l}(g(k,l)-\bar{g})^2 \sum_{k,l}(f(k,l)-\bar{f})^2}}, \tag{2}$$

where $\bar{g}$ and $\bar{f}$ are the mean intensity values of patches g and f. The values of NCC are always between -1 and 1, and the larger the value the more similar the patches are.

b) How many correct correspondences do you get by using NCC instead of SSD?

c) Which one of the two similarity measures performs better in this case and why?

Exercise 2. Matching SURF regions.
Run the example file `SURFmatching.m` which extracts and matches SURF interest regions by using the functionality provided by Matlab's Computer Vision System toolbox. SURF is quite similar to SIFT which was presented in Lecture 3. In this implementationt the descriptor vectors for the local regions have 64 elements (instead of 128 in SIFT) but Euclidean distance can still be used as a similarity measure in descriptor space. See the comments in the source code and do the following tasks:
a) Sort the given nearest neighbor matches in ascending order based on the *nearest neighbor distance ratio* (NNDR), which is defined in Equation (4.18) in the course book. Report the number of correct correspondences among the top 5 matches based on NNDR and compare it to the case where ordering is based on nearest neighbor distance. (Since NNDR is not symmetric, you can find the nearest and second nearest match from image 2 for each feature of image 1, or vice versa.)
b) Give justified answers: What are the benefits of using SIFT/SURF regions instead of Harris corners? Why the matching approach of task 1 (i.e. Harris corners and NCC based matching) would not work for the example images of task 2? In what kind of cases Harris corners may still be more suitable than SURF and why?

Exercise 3. Scale-space blob detection.
Run the example script `evaluateScaleSpaceBlobs.m` which illustrates pre-computed blob detections obtained with a similar procedure as implemented in SIFT and described below. Here the task is to replace the pre-computed regions with regions computed by your own implementation. The result does not need to be exactly the same as the pre-computed one but similar. In summary, fill in the short missing part to `scaleSpaceBlobs.m` in order to implement the scale-space blob detector defined by the following steps:
a) Generate a Laplacian of Gaussian filter. (You can set $\sigma = 0.5$.)
b) Build a Laplacian scale space, starting with some initial scale and going for n iterations:
- **filter image with scale-normalized Laplacian at current scale**
- save square of Laplacian response for current level of scale space
- increase scale by factor k

c) Perform non-maximum suppression in scale space.
d) Display resulting circles at their characteristic scales.
(Hint: Note that only the part indicated with bold text above is missing from the example code.)

Apply the blob detector to example images `boat1.png` and `boat6.png` as shown in the example script. Can you identify some corresponding regions in the visualized results?

Note 1: Suitable values for k and n could be $k = 1.19$ and $n = 18$.
Note 2: This task corresponds to Exercise 4.1 in the course book. A similar assignment has been used by Lazebnik at UIUC and their course page gives also more detailed instructions: `http://slazebni.cs.illinois.edu/spring16/assignment2.html`.