

# DATA.ML.300 Computer Vision

## Exercise Round 4

For these exercises you will need Python. Return all your answers and output figures in a pdf-file. In addition, submit runnable .py files, where you have filled in your codes to the template files. Do not include all the code from the .py files in the pdf. Exercise points will be granted after a teaching assistant has checked your answers. Returns done before the deadline will result in maximum of 4 points, whereas returns after the deadline will result in maximum of 1 point.

**Task 1.** Hough transform and parameter spaces (Pen & paper) (1 point)

a) We have two pixels of an image in the (x, y)-plane:  $(x_0, y_0) = (-2, 1)$  and  $(x_1, y_1) = (2, 5)$ . Plot the lines that these points create in Cartesian parameter space. What is the point of intersection  $(m', b')$ ?

b) Plot the sinusoids that these points create in Polar coordinate parameter space. What is the point of intersection  $(\theta', \rho')$  when  $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$ ?

c) What advantages does the Polar coordinate form have over Cartesian coordinate form?

**Include your answers and the plots to your pdf-file.**

**Task 2.** Robust line fitting using RANSAC. (Programming exercise) (1 point)

Open `RobustLineFitting.py`, which plots a set of points  $(x_i, y_i), i = 1, \dots, n$ , and estimate a line that best fits to these points by implementing a RANSAC approach as explained in the lecture slides. Your task is to implement the missing code stated in the comments and observe the results. **Include your answers and the output plot to your pdf-file, and return also your runnable version of RobustLineFitting.py.**

**Task 3.** Matching Harris corner points. (Programming exercise) (1 point)

Open `HarrisMatching.py` 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

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

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

Do the task (a) below and answer question (b). **Include your answers and the output plots to your pdf-file, and return also your runnable version of HarrisMatching.py**

a) Implement the matching of mutually nearest neighbors using normalized cross-correlation (NCC) as the similarity measure instead of SSD. For two image patches of similar size NCC is defined as follows:

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

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

**Task 4.** Matching SURF regions. (Programming exercise) (1 point)

Open `SURFmatching.py` and see the instructions in the comments of the source code. The code loads pre-calculated SURF-feature vectors for matching SURF interest regions. SURF is quite similar to SIFT which was presented in lecture slides. In this implementation 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. Your task is to implement the missing code stated in the comments and do the following tasks. **Include your answers and the output plots to your pdf-file, and return also your runnable version of SURFmatching.py.**

a) Sort the given nearest neighbor matches in ascending order based on the *nearest neighbor distance ratio* (NNDR), which is defined

$$NNDR = \frac{d_1}{d_2} = \frac{\|D_A - D_B\|}{\|D_A - D_C\|}$$

where  $d_1$  and  $d_2$  are the nearest and second nearest neighbor distances,  $D_A$  is the target descriptor, and  $D_B$  and  $D_C$  are its closest two neighbors.

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.

b) What are the benefits of using SURF regions instead of Harris corners? In what kind of cases Harris corners may still be better than SURF and why?