Exercises for
Pattern Analysis
Daniel Stromer & Dalia Rodriguez Salas
Assignment 1, 16/17.05.2018

General Information:

Lecture (3 SWS):	Tue 12.15 – 13.45 (H16) and Thu 12.15 – 13.45 (H16)
Exercises (1 SWS):	Tue 14.00 – 16.00 (02.151b-113) and Thu 10.00 – 12.00 (02.151b-113)
Certificate:	Oral exam at the end of the semester
Contact:	daniel.stromer@fau.de & dalia.rodriguez@fau.de

Manifold Learning

Exercise 1 Rules for submitting the programming exercise:

(a) Work together in pairs (max. two people).

(b) You have to show your code to the tutors not later than the deadline. It's is recommended that all team members shows up.

(c) Your code has to be in C or C++. We recommend using *OpenCV* for Matrix algebra and visualization, as it is available in the CIP pool.

(d) You can either use CIP pool PC's or your own laptop.

(e) Plagiarism will be punished by assigning zero points, removal from the programming exercises and/or by a report to the examination office. According to Wikipedia, plagiarism is *the "wrongful appropriation" and "stealing and publication" of another author's "language, thoughts, ideas, or expressions" and the representation of them as one's own original work.*

Exercise 2 Programming exercise:

The goal of this exercise is to implement two algorithms for manifold learning: Principle Component Analysis (PCA) and Isomap. We will apply these algorithms to 3-D data to reduce it to 2-D data.

(a) Download *main.cpp* from Studon. This file contains code that generates sample 3-D data, as well as functions for visualization.

(b) Implement Principle Component Analysis (PCA) in the given function body. Scale the data such that the L2 norm in each dimension is one.

(c) Implement Isomap in the given function body. Use k-NearestNeighbor and Euclidean distances for the neighborhood search. Using the neighbors, create a distance map using, e.g., the Floyd-Warshall algorithm. Obtain the manifold embedding by eigen-decomposition. We recommend using *10* neighbors, and a default distance of *10000.0* for samples which are not neighbors to ensure a path between all sample combinations.

(d) Enable the commented code in the *main()* function to display and save the results of your algorithms. Results obtained with the reference implementation are available from StudOn for comparison.

(e) (not graded) Investigate the behavior of the algorithms when changing the *noiseScaling*.

(f) **Deadline for submission: May, 29/30th**