Analysis of Three-Dimensional Shapes
F. R. Schmidt, M. Vestner, Z. Lähner
Summer Semester 2017

Computer Vision Group
Institut für Informatik
Technische Universität München

Weekly Exercises 6

Room: 02.09.023
Wed, 28.06.2017, 14:00-16:00
Submission deadline: Tue, 27.06.2017, 23:59 to laehner@in.tum.de

Programming: Multi-Dimensional Scaling

Download the new supplementary material. It contains an outline for the whole exercise and code for several side tasks as well as visualizing the solution as well as a new 3D shape with a corresponding distance matrix.

Exercise 1 (6 points)**.** Implement the Multi-Dimensional Scaling approach to find a correspondence between two shapes. `ex6_1.m` already contains an outline for the whole procedure and you can fill in code into `mds.m`, `alignpoints.m` and `extract_matching.m`.

1. `mds.m` should contain the algorithm as explained in the lecture. It takes a distance matrix $D \in \mathbb{R}^{n \times n}$ and a dimension to embedd in $m \in \mathbb{N}$. It should return the coordinates of each point in $\mathbb{R}^m$ as a matrix $Z \in \mathbb{R}^{n \times m}$. A distance matrix for two different shapes is included in the supplementary material. There are two instances of each type and their vertices are ordered in the same way, so the distances are valid for both of them. The parameters `epsilon` and `maxI` control the minimum relative progress (if you don't know how to compute it, just skip it) and the maximum number of iterations.

2. `alignpoints.m` should align two point clouds $Z_1, Z_2 \in \mathbb{R}^m$ with rigid transformations (i.e. translation and rotation). The result should be two point sets $\hat{Z}_1 = \{R_1(z + t_1) | z \in Z_1\}, \hat{Z}_2 = \{z + t_2) | z \in Z_2\}$ that were aligned by the optimal rigid transformations.

   The translations t_1, t_2 can be found by computing the point clouds mean. For the rotation we suggest to align the principal axes $a_1, ..., a_m \in \mathbb{R}^m$ with the standard Euclidean axes. Note that since the signs of the principal axes are not uniquely determined, the visualization code includes sign parameters that can be altered manually.

3. `extract_matching.m` For each $z \in \hat{Z}_1$ find the $z' \in \hat{Z}_2$ that is the nearest neighbor. This will give you a matching (z, z') for each point in $\hat{Z}_1$. Notice that using this procedure will not necessarily give you a bijection between both shapes. You can use `visualize_matching.m` to visualize your results.

The Matlab functions `mean` and `pca` might be helpful, for nearest neighbor search you can use `knnsearch`.