Analysis of Three-Dimensional Shapes
F. R. Schmidt, M. Vestner, Z. Lähner
Summer Semester 2016

Computer Vision Group
Institut für Informatik
Technische Universität München

Weekly Exercises 3

Room: 02.09.023
Wed, 18.05.2016, 14:00-16:00 (in two weeks!)
Submission deadline: Tue, 17.05.2016, 23:59 to laehner@in.tum.de

Mathematics: Linear Algebra Recap

Exercise 1 (2 points)**.** Consider two parametrizations of an arc

$$f_i : [0,1] \to \mathbb{R}^2$$
$$f_1 : t \mapsto \begin{pmatrix}\cos(t)\\ \sin(t)\end{pmatrix}, \qquad f_2 : t \mapsto \begin{pmatrix}\cos(t^2)\\ \sin(t^2)\end{pmatrix}$$

And the function

$$s : \mathbb{S}^1 \to \mathbb{R}$$
$$(x,y) \mapsto y$$

Calculate the integrals $\int_0^1 (s \circ f_i)(t)dt$ and compare the results to the line integrals.

Exercise 2 (1 point)**.** Consider a sequence of values $x_i \in \mathbb{R}, i \in \{1, ..., n\}$. In the lecture we saw that $\text{mean}(x_i) = \sum_i \frac{x_i}{n} = \arg\min_x \sqrt{\sum_i (x_i - x)^2}$ based on the l_2-norm. Derive an explicit formulation based on the l_1-norm, i.e. rewrite $\arg\min_x \sum_i |x_i - x|$ without a arg minimum.

The following exercise is related to the lecture on May 9th and probably not solvable beforehand.

Exercise 3 (4 points)**.** Let $M = (\mathcal{V}, \mathcal{T})$ be a manifold given by a triangle mesh and f, g piecewise linear functions on M given by their coefficients $\alpha, \beta \in \mathbb{R}^n$ in the hat basis$\{\psi_1, \ldots, \psi_n\}$. Remember that each triangle T_i is mapped by its own coordinate map x_i and $\int_M f(x)\ dx = \sum_i \int_{T_i} f(y)\ dy$.

1. Find a matrix A and vector α such that $A\alpha = \int_M f(x)\ dx$.
2. Rewrite $\langle f, g \rangle_M = \int_M f(x)g(x)\ dx$ as a matrix-vector product $\alpha^\top M \beta$. We call M the mass matrix.
3. Show that the mass matrix is symmetric and positive definite. With this properties we know that it defines an inner product $\langle a, b \rangle_M = a^\top M b$ (and therefore also induces a norm).

Programming: Shape Matching and 3D Shapes

We will consider the same images as in the last exercise, download the supplementary material 02 from the website if you have not done so already. Additionally there is new supplementary material containing a 3D Shape, a function `extract_pointwise_contour.m` to extract a contour as a sequence of 2D coordinates, `LAP.m` solving Linear Assignment Problems (actually not with the Hungarian method...) and `visualise_matching.m`.

Exercise 4 (2 points)**.** Extract pointwise contours of the images `bat-9.gif`, `device7-1.gif` and `turtle-1.gif` with 100 points.

1. Calculate the integral invariant on each image and evaluate them at the pointwise contour. Try gaussians kernel with sizes of 10×10, 32×32, 100×100 and 200×200. (For the std deviations 3, 10, 30, 80) Include figures of your results in your solution sheet (try `scatter` with colors for the pointwise contour and `fspecial`, `conv2`, `interp2` for the feature).

2. Calculate the shape context on the images with a 101×101 kernel divided into 3 different radii and 10 different angles (in the lecture the kernel had 2 radii and 3 angles). This means you have to produce 30 different kernels and your feature at each point on the contour will be a $\mathbb{R}^{30}$ vector.

Exercise 5 (2 points)**.** Calculate the best matching between the pointwise contours of `turtle-1.gif` to `turtle-19.gif` and `apple-20.gif`. Create the 3 cost matrices for the linear assignment problem with the 3 different features (curvature, integral invariant, shape context) and the distance functions proposed in the lecture. Choose the parameters that you think will work the best. Then solve for the permutation with `LAP.m` from the supplementary material. You can visualise your results with `visualise_matching.m` giving both pointwise contours and your permutation as an input. Points with the same color are matched to each other. Are the matchings reasonable?

The following exercise is mostly related to the lecture on May 9th (but it is possible to do it beforehand).

Exercise 6 (1 point)**.** We represent 3D shapes as triangle meshes $\mathcal{M} = (\mathcal{V}, \mathcal{T})$ embedded in $\mathbb{R}^3$. They consist of vertices $\mathcal{V} = \{v_1, ..., v_n\}$ and triangles $\mathcal{T} \subset \mathcal{V} \times \mathcal{V} \times \mathcal{V}$ connecting them to a closed surface. The coordinates of each vertex are denoted by $x(v_i) \in \mathbb{R}^3$. Note that the vertices of a triangle $t = (u, v, w) \in \mathcal{F}$ are ordered. We define triangles to be identical if they can be transformed into each other by a cyclic permutation, i.e. $(u, v, w) = (v, u, v) = (w, u, v)$ but $(u, v, w) \neq (w, v, u)$. The order defines the direction of the normal and is also called orientation.

1. There are different formats to store triangle meshes. The easiest one is the vert-tri format consisting of two files, one containing the 3D coordinates and the other one the triangles. The file `name.vert` contains n rows with three double numbers separated by tabs. Each row represents one 3D coordinate.

The file `name.tri` (note that the file name has to be the same as for the .vert) contains m rows with three positive integer values $x \leq n$. Each x refers to a row/vertex from the .vert file. The order of x in each row defines the orientation of the triangle. Write a function `read_vert_tri.m` that takes a filename and returns a struct `M` with fields `M.VERT` a $n \times 3$ array and `M.TRIV` a $m \times 3$ array. You can try your function on the cat0 files from the supplementary material (and see if you did everything correctly in the next exercise).

2. Triangulated surfaces can be plotted in Matlab with `trisurf(TRI, X, Y, Z)`. You need to give each column of `M.VERT` separatly to the function. The colors you see on the surface represent the height function.

3. Implement a function `facearea` that takes a triangle mesh struct as an input and returns a $\mathbb{R}^m$ vector containing the area of each triangle. You can plot your function by using `trisurf(TRI, X, Y, Z, f)` where f is a vector with as many entries as the numbers of vertices or faces. (Of course the entries in the vector must be ordered in the same way as the vertices or faces they correspond to.)

4. Implement a function `mass_matrix.m` that takes a triangle mesh and returns a sparse diagonal matrix with the mass of each vertex. As a reminder, $m_i = \sum_{j \in \mathcal{N}(i)} \frac{\text{area}(j)}{3}$, i.e. the sum of the areas of all faces next to vertex i divided by 3.