Computer Vision Group
Institut für Informatik
Technische Universität München

Weekly Exercises 5

Room: 02.09.023

Wed, 01.06.2016, 14:00-16:00

Submission deadline: Tue, 31.05.2016, 23:59 to laehner@in.tum.de

Mathematics

Exercise 1 (2 points)**.** Find a map $\varphi : T_{\text{ref}} \to \mathbb{R}^3$ that is

1. angle-preserving but not area-preserving
2. area-preserving but not angle-preserving

$T_{\text{ref}} = \operatorname{conv}((0,0),(0,1),(1,0))$ is the reference triangle as used in the lecture. The image should be a triangle in 3D.

Solution. 1.

$$\begin{aligned} &\varphi_1 : (u,v) \mapsto (2\cdot u, 2\cdot v, 0) \\ &g = \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix} \end{aligned}$$

2.

$$\begin{aligned} &\varphi_2 : (u,v) \mapsto u \cdot \begin{pmatrix} \frac{1}{\sqrt[4]{3/4}} & 0 & 0 \end{pmatrix} + v \cdot \begin{pmatrix} \frac{1}{2\cdot\sqrt[4]{3/4}} & \sqrt[4]{3/4} & 0 \end{pmatrix} \\ &a = \sqrt[4]{3/4} \\ &g = \begin{pmatrix} \frac{1}{a^2} & \frac{1}{2a^2} \\ \frac{1}{2a^2} & \frac{1}{4a^2} \end{pmatrix} \end{aligned}$$

Exercise 2 (3 points)**.** The stiffness matrix $C \in \mathbb{R}^{n\times n}$ was defined in the lecture as $C_{ij} = \int_S \langle \nabla\phi_i(x), \nabla\phi_j(x)\rangle \mathrm{d}x$. Show that the entries are equal to

$$C_{ij} = \begin{cases} -\frac{\cot(\alpha_{ij})+\cot(\beta_{ij})}{2} & \text{if } (i,j) \text{ an edge} \\ -\sum_{k\neq i} C_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

The derivation is similar to the one of the mass matrix shown in Exercise Sheet 3.

Solution. Let $\tilde{\phi}_i(p) = \phi_i(x_k(p))$ and $e_1 = x_k(u_1)$, $e_2 = x_k(u_2)$.

First case, $i \neq j$:

$$\begin{aligned} C_{ij} &= \int_S \langle \nabla \phi_i(x), \nabla \phi_j(x) \rangle \mathrm{d}x \\ &= \sum_{k \in \mathcal{T}} \int_{T_k} \langle \nabla \phi_i(p), \nabla \phi_j(p) \rangle \mathrm{d}p \end{aligned}$$

$$\begin{aligned} \int_{T_k} \langle \nabla \phi_i(p), \nabla \phi_j(p) \rangle \mathrm{d}p &= \int_{T_{\text{ref}}} \langle g_k^{-1} \nabla \tilde{\phi}_i(p), g_k^{-1} \nabla \tilde{\phi}_j(p) \rangle \sqrt{det(g_k)} \mathrm{d}p \\ &= \sqrt{det(g_k)} \begin{pmatrix} 1 & 0 \end{pmatrix} g_k^{-1} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \underbrace{\int_{T_{\text{ref}}} 1 \; \mathrm{d}p}_{=1/2} \\ &= \frac{1}{2} \frac{\sqrt{det(g_k)}}{det(g_k)} g_k^{12} \\ &= \frac{1}{2} \frac{-\langle e_1, e_2 \rangle}{\sqrt{det(g_k)}} \\ &= -\frac{1}{2} \frac{||e_1|| \cdot ||e_2|| \cdot \cos(\alpha_{ij})}{2 \cdot \text{area}(T_k)} \\ &= -\frac{1}{2} \frac{||e_1|| \cdot ||e_2|| \cdot \cos(\alpha_{ij})}{||e_1|| \cdot ||e_2|| \cdot \sin(\alpha_{ij})} \\ &= -\frac{\cot(\alpha_{ij})}{2} \end{aligned}$$

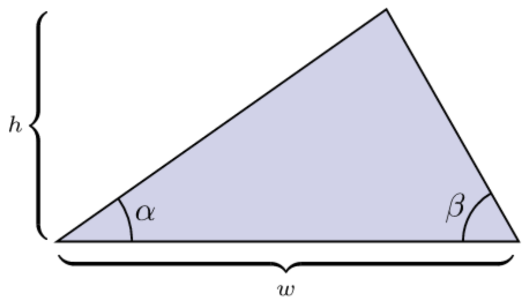
The summands are only non-zero at triangles adjacent to both i and j. See the lecture slides for a sketch of α_{ij}, β_{ij}.

$$\begin{aligned} C_{ij} &= \sum_{k \in \mathcal{N}_i \cap \mathcal{N}_j} -\frac{\cot \alpha_{ij}}{2} \\ &= -\frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} \end{aligned}$$

Second case, $i = j$:

$$\begin{aligned} C_{ii} &= \int_S \langle \nabla \phi_i(x), \nabla \phi_i(x) \rangle \mathrm{d}x \\ &= \sum_{k \in \mathcal{T}} \int_{T_k} ||\nabla \phi_i(p)||^2 \mathrm{d}p \end{aligned}$$

$$
\begin{aligned}
\int_{T_k} ||\nabla \phi_i(p)||^2 \mathrm{d}p &= \int_{T_{\text{ref}}} ||g_k^{-1} \nabla \tilde{\phi}_i(p)||^2 \sqrt{det(g_k)}\ \mathrm{d}p \\
&= \sqrt{det(g_k)} \begin{pmatrix} 1 & 0 \end{pmatrix} g_k^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \int_{T_{\text{ref}}} 1\ \mathrm{d}p \\
&= \frac{||e_1||^2}{4 \cdot \text{area}(T_k)} \\
&= \frac{w^2}{2 \cdot w \cdot h} \\
&= \frac{w}{2 \cdot h} \\
&= \frac{1}{2}(\cot(\alpha) + \cot(\beta))
\end{aligned}
$$



e_1 is the edge opposing vertex i in each triangle. We can write the diagonal entries as sums over triangles or one can see that the angles showing up are exactly the same as in the entries C_{ij} but paired up differently (see below).

$$
\begin{aligned}
C_{ii} &= \sum_{k \in \mathcal{N}_i} \frac{\cot \alpha_k + \cot \beta_k}{2} \\
&= \sum_{(i,j) \text{ edge at } i} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} = \sum_j C_{ij}
\end{aligned}
$$

Programming

Exercise 3 (4 points)**.** This exercise will contain the first steps for implementing the gradient on triangle meshes $(\mathcal{V}, \mathcal{T})$. The gradient ∇f of a function $f : S \to \mathbb{R}$ can be calculated by taking $\nabla f = g^{-1} \cdot \nabla \tilde{f}$ where g is the first fundamental form and, for a fixed coordinate map x of S, $\tilde{f} : U \to \mathbb{R}$ is such that $f = \tilde{f} \circ x^{-1}$. Since $g = (Dx)^\top Dx$, we start with calculating Dx and then g.

1. Remember we have a coordinate map for each triangle individually, but instead of being given the map x_k for each triangle k we only have the vertex coordinates. Think about how each x_k and Dx_k looks like. (Tip: They were already used in Exercise 3.) Implement a function `trimesh_differential` that takes a triangle mesh and returns a $\mathbb{R}^{3 \times 2 \times k}$ multi-dimensional array representing all differentials.

2. The first fundamental form is constant on each triangle, we can represent it as a $\mathbb{R}^{2\times2\times k}$ matrix. Write a function `trimesh_fff` that takes the result of `trimesh_differential` and returns the first fundamental form as a tensor. In theory multiplication of matrices with more dimensions works the same way as with two, but it is not implemented in Matlab so you will have to simulate it with a for loop. The function `squeeze` will help to return to matrices when taking slices of the tensor.

3. Calculate the area of each triangle with the first fundamental form and compare your results to the areas you calculated in Exercise 3 (which were probably done with Heron's formula).