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 3

Room: 02.09.023
Wed, 07.06.2017, 14:00-16:00
Submission deadline: Tue, 06.06.2017, 23:59 to laehner@in.tum.de

Mathematics: Parametrization

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

$$c_i : [0,1] \to \mathbb{R}^2$$
$$c_1 : t \mapsto \begin{pmatrix} \cos(t) \\ \sin(t) \end{pmatrix}, \qquad c_2 : t \mapsto \begin{pmatrix} \cos(t^2) \\ \sin(t^2) \end{pmatrix}$$

And the function

$$\begin{aligned} f :& \mathbb{S}^1 \to \mathbb{R} \\ & (x,y) \mapsto y \end{aligned}$$

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

Solution. Normal integrals:

$$\int_0^1 (f \circ c_1)(t)\mathrm{d}t = \int_0^1 \sin(t)\mathrm{d}t = 0.46$$
$$\int_0^1 (f \circ c_2)(t)\mathrm{d}t = \int_0^1 \sin(t^2)\mathrm{d}t = 0.3$$

Line Integral:

$$\int_\Gamma f(s)(d)s = \int_0^1 (f \circ c_i)(t) \cdot ||c_i'(t)||\mathrm{d}t$$
$$\begin{aligned} \int_0^1 (f \circ c_1)(t) \cdot ||c_1'(t)||\ \mathrm{d}t &= \int_0^1 f(c_1(t)) \cdot ||\begin{pmatrix} -\sin(t) \\ \cos(t) \end{pmatrix}||\mathrm{d}t \\ &= \int_0^1 \sin(t)\sqrt{\sin(t)^2 + \cos(t)^2}\mathrm{d}t = 0.46 \\ \int_0^1 (f \circ c_2)(t) \cdot ||c_2'(t)||\ \mathrm{d}t &= \int_0^1 f(c_2(t)) \cdot ||\begin{pmatrix} -2t\sin(t^2) \\ 2t\cos(t^2) \end{pmatrix}||\mathrm{d}t \\ &= \int_0^1 \sin(t^2)\sqrt{4t^2\sin(t^2)^2 + 4t^2\cos(t^2)^2}\mathrm{d}t = 0.46 \end{aligned}$$

Exercise 2 (2 points)**.** Show that the push-forward is a linear mapping.

Solution. For a linear mapping we have to show that

$$\begin{aligned} Df(p)[v_1+v_2] &= Df(p)[v_1] + Df(p)[v_2] \\ Df(p)[\lambda v] &= \lambda Df(p)[v] \end{aligned}$$

when $f : M \to N$.

We will use the definition of the push-forward using curves but the proof can as easily (maybe even easierly) with the differentials of x and f. Let $x : U \to M$ be a coordinate map w.l.o.g. $0 \in U \subset \mathbb{R}^d$ and $x(0) = p$ as well as $f(p) = q$. Let $u, v \in T_pM$ then there exist $h_u, h_v \in \mathbb{R}^d$ such that

$$\begin{aligned} c_u :\ & (-\epsilon, \epsilon) \to M & t &\mapsto x(t \cdot h_u) \\ c_v :\ & (-\epsilon, \epsilon) \to M & t &\mapsto x(t \cdot h_v) \end{aligned}$$

They define the equivalence classes $[c_u] = u$ and $[c_v] = v$ for which hold that $c_{u/v}(0) = p$ and $Dc_{u/v}(0) = u/v$. Further, we have

$$\begin{aligned} f \circ c_u :\ & (-\epsilon, \epsilon) \to N \\ f \circ c_v :\ & (-\epsilon, \epsilon) \to N \end{aligned}$$

and it follows by definition that $[f \circ c_u] \in T_qN$ and $[f \circ c_v] \in T_qN$. We define $[c_u] + [c_v] = [c_{u+v}]$ by

$$\begin{aligned} c_{u+v} &: (-\epsilon, \epsilon) \to M \\ & \quad t \mapsto x(t \cdot (h_u + h_v)) \\ c_\lambda &: (-\epsilon, \epsilon) \to M \\ & \quad t \mapsto x(t \cdot \lambda h_u) \end{aligned}$$

$$\begin{aligned} Df(p)[c_{u+v}] &= [f \circ c_{u+v}] \\ &= [(f \circ x)(t \cdot (h_u + h_v))] \\ &= \frac{\partial}{\partial t}((f \circ x)(t \cdot (h_u + h_v)))\ |_{t=0} \\ &= \frac{\partial (f \circ x)}{\partial t}(t \cdot (h_u + h_v)) \cdot (h_u + h_v)\ |_{t=0} \\ &= \frac{\partial (f \circ x)}{\partial t}(0) \cdot h_u + \frac{\partial (f \circ x)}{\partial t}(0) \cdot h_v \\ &= [f \circ c_u] + [f \circ c_v] \\ &= Df(p)[c_u] + Df(p)[c_v] \end{aligned}$$

and

$$
\begin{aligned}
Df(p)[c_\lambda] &= [f \circ c_\lambda] \\
&= [(f \circ x)(t \cdot \lambda h_u)] \\
&= \frac{\partial}{\partial t}(f \circ x)(t \cdot \lambda h_u) \mid_{t=0} \\
&= \frac{\partial (f \circ x)}{\partial t}(t \cdot \lambda h_u) \cdot \lambda h_u \mid_{t=0} \\
&= \lambda \cdot \frac{\partial (f \circ x)}{\partial t}(0) h_u \\
&= \lambda [f \circ c_u] \\
&= \lambda \cdot Df(p)[c_u]
\end{aligned}
$$

Programming: 2D Shape Features

Download the supplementary material from the homepage. It contains some black-white silhouette images from the MPEG7 dataset (http://www.dabi.temple.edu/˜shape/MPEG7/dataset.html), 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`. Please also submit your code.

Exercise 3 (2 points)**.** Read out the image files `bat-9.gif`, `device7-1.gif`, `turtle-1.gif` into matrices (use `imread`, it reads positive integers. Changing the type to double will help). Include an image for each sub-exercise in your solution sheet.

1. Calculate the curvature on the contour. It can be seen as the level set function somewhere between 0 and 1 so the formula $\kappa(p) = \text{div}\left(\frac{\nabla F(\cdot)}{||\nabla F(\cdot)||}\right)(p)$ from the lecture can be used. There are `imgradient` and `imgradientxy` in Matlab or implement your own finite difference gradient as an exercise (it's quite easy). The divergence can be calculated with `divergence`.

2. The result from the last exercise was pretty ugly. The reason is that the function we considered was not smooth but went zig-zag along the edges of the pixels. Use a gaussian filter on the image before calculating the curvature. (See `imgaussfilt`) Use the `extract_contour.m` from the supplementary material to get a binary mask for the contour with different thickness. Play around with different σ for the filter and thicknesses of the mask.

Exercise 4 (1 point)**.** In most applications we want to find out which shapes are similar to each other. Create a descriptor for each shape in the supplementary material and create a histogram of the different curvatures on the contour (don't forget to normalize because normally the contours will not have the same amount of points). There is a Matlab function `histogram` if you are not familiar with histograms.

Compare the descriptor of `device7-1` to all other descriptors (for example with the Euclidean distance) and sort the remaining shapes in order of similarity to device7-1.

Exercise 5 (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 but this is not enough for real applications). 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 6 (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?