Digital 3D Geometry Processing
Exercise 2 – Curves

Handout date: 04.10.2016
Submission deadline: 13.10.2016, 23:00 h

What to hand in

A .zip compressed file renamed to `Exercise`*n*`-GroupMemberNames.zip` where *n* is the number of the current exercise sheet. It should contain:

- Hand in **only** the files you changed (headers and source). It is up to you to make sure that all files that you have changed are in the zip.
- A `readme.txt` file containing a description on how you solved each exercise (use the same numbers and titles) and the encountered problems.
- Other files that are required by your `readme.txt` file. For example, if you mention some screenshot images in `readme.txt`, these images need to be submitted too.
- For the theory exercise, put `TheoryExercise.txt` with your solutions in the same .zip you submit for the code.
- Submit your solutions to Moodle before the submission deadline. Late submissions will receive 0 points!

The Framework

To load the project, open the CMakeLists.txt from the main directory using Qt Creator. In order to compile the code, you need to have some external libraries installed in your system.

- On Ubuntu (15.04, 15.10, ...) follow these insturctions

```
~$ sudo apt-get install libglew libglew-dev
~$ sudo apt-get install libglfw3 libglfw3-dev
```

- On Mac OSX (10.10.2, 10.11, ...) install `homebrew` from `http://brew.sh` and follow these instructions

```
~$ brew install glfw3
~$ brew install glew
```

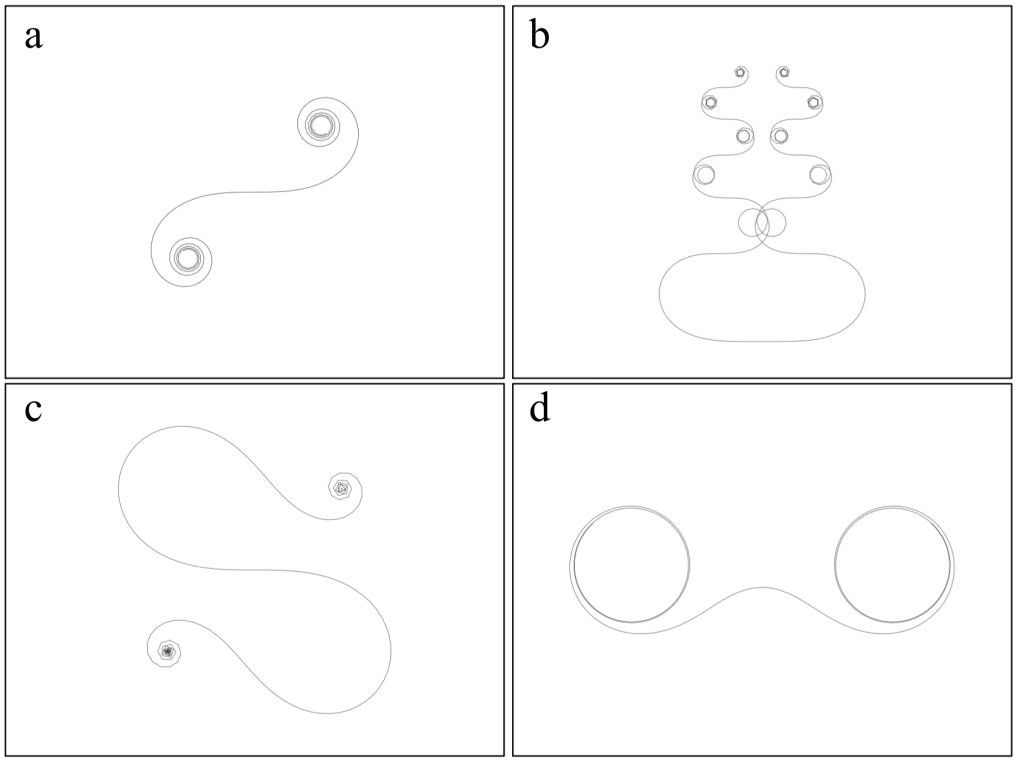

Figure 1: Curves reconstructed from given curvature functions $k_i(s)$.

- On Windows we provide the binaries for `glfw3` (tested with Visual Studio 2015 compiler) and `glew`. If you encounter linking errors, you might need to build `glfw` and `glew` on your own. Download the source files from GLFW3 and GLEW, build, and copy the binaries to the folder: `dgp2016-exercise2/externals/{glfw3, glew}/lib`. We provide `glfw3` binaries for Visual Studio Compiler 10, 11, 12 and 14, you can change lines 28 and 31 in `dgp2016-exercise2/cmake/FindGLFW3.cmake` to match with your version.

For the exercise 2.1 you will need to fill in the missing code in the file `curves.cpp`. For the exercise 2.2 you will need to add your solution in the file `reconstruction.cpp`. Compiling the project will produce two executable files named `smoothing` and `reconstruction`. For smoothing use keys `S` to `C` to iterate and for reconstruction use `SPACE` to compute a new point of the curve.

1 Theory Exercise

Match each curve from Figure 1 with the corresponding curvature function:

$$k_1(s) = \frac{s^2 - 1}{s^2 + 1}, \quad k_2(s) = s, \quad k_3(s) = s^3 - 4s, \quad k_4(s) = \sin(s)s. \tag{1}$$

Write your solutions in a file named `TheoryExercise.txt`. If, for example, the curvature function $k_1(s)$ corresponds to the curve **a**, write **1-a** as your solution. Briefly explain your answers.

2 Coding Exercise

2.1 Curve Smoothing

Given a curve in $\mathbb{R}^2$ as a closed polyline with vertices $\{V_i\}_{i=1}^{M}$, where $V_1 = V_M$, implement the following two examples of curve smoothing:

- Move every vertex towards the centroid of the two neighbors. More specifically, every vertex V_i should shift to vertex V_i' according to

$$V_i' = (1-\varepsilon)V_i + \varepsilon\frac{V_{i-1}+V_{i+1}}{2} \tag{2}$$

  where ε is a small time step (you can experiment with different values for the time step). Iterate this procedure. After each iteration uniformly scale the curve to its original length.
  Implement this smoothing by filling in the laplacianSmoothing() function in the file `curves.cpp`. This function should perform one iteration of smoothing. Press key `S` to run this function and iterate.

- Move every vertex towards the center of the osculating circle. Consider an osculating circle at vertex V_i as a circumscribed circle O of a triangle defined with vertices V_{i-1}, V_i and V_{i+1}. This can be done by shifting every vertex V_i to vertex V_i' according to

$$V_i' = V_i + \varepsilon\,\frac{C - V_i}{\|C - V_i\|^2} \tag{3}$$

  where ε is a small time step and C is the center of the circumscribed circle O. Iterate this procedure. After each iteration uniformly scale the curve to its original length.

  Implement this smoothing by filling in the osculatingCircle() function in the file `curves.cpp`. This function should perform one iteration of smoothing. Press key `C` to run this function and iterate.

2.2 Reconstruction of Curve

Let a curve $\boldsymbol{c}(s) = [x(s), y(s)]$ be parametrized by arc length in $\mathbb{R}^2$. The derivative of $\boldsymbol{c}(s)$ with respect to arc length s, $\boldsymbol{t}(s) = \frac{d\boldsymbol{c}(s)}{ds} = [\cos\varphi(s), \sin\varphi(s)]$ is a unit vector tangent to the curve, where $\varphi(s)$ denotes the angle between the tangent at a point $[x(s), y(s)]$ and the positive direction of the x-axis (see Figure 2). The curvature at a point measures the rate of curving (bending) as the point moves along the curve with unit speed and can be defined as

$$k(s) = \frac{d\varphi(s)}{ds}. \tag{4}$$

The curvature of a curve determines the curve up to rigid transformations. One can reconstruct the curve from its curvature by integration

$$\varphi(s) = \int k(s)ds + \varphi_0, \quad \boldsymbol{c}(s) = [\int \cos\varphi(s)ds + x_0, \int \sin\varphi(s)ds + y_0], \tag{5}$$

where φ_0, x_0, y_0 are constants of integration.

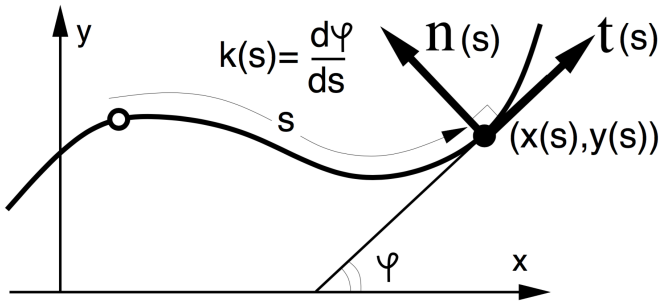

Figure 2: Definition of curvature.

Given a curvature function $k(s)$, a tangent vector $t(0)$ of the curve $\boldsymbol{c}(s)$ at the time 0 and the starting point $\boldsymbol{c}(0) = [x_0, y_0]$ of the curve $\boldsymbol{c}(s)$, your task is to compute a discrete approximation of the curve $c(s)$ in $\mathbb{R}^2$.

Implement the reconstruction of curves by numerical integration where the following is given:

- curvature $k(s) = 1$, $\boldsymbol{c}(0) = [0, -1]$, $\boldsymbol{t}(0) = [1, 0]$
- curvature $k(s) = s$, $\boldsymbol{c}(0) = [0, 0]$, $\boldsymbol{t}(0) = [1, 0]$
- curvature $k(s) = s^2 - 2.19$, $\boldsymbol{c}(0) = [0, 0]$, $\boldsymbol{t}(0) = [1, 0]$

Experiment with different time steps.

To implement the reconstruction filling in the reconstructCurveStep() function in the file `reconstruction.cpp`. This function should perform one iteration of integration. Press `SPACE` key to run this function and iterate. Three output curves are shown in Figure 3.

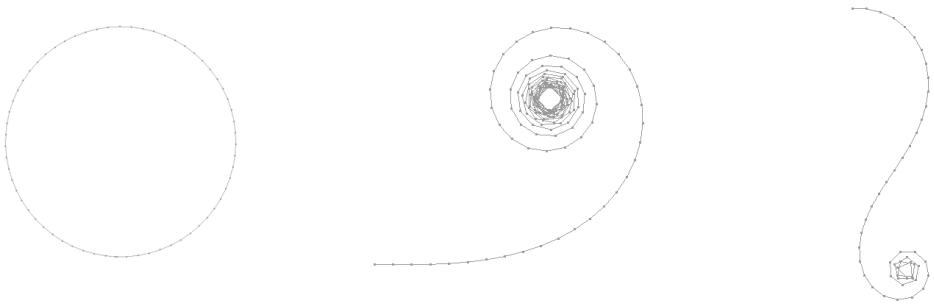

Figure 3: Three output curves for exercise 2.2.