19 May 2019

# General Information:

You have to submit your solution via Moodle. We allow **groups of two students**. Please list all names in the submission comments, upload one solution per group. You have **two weeks of working time** (note the deadline date in the footer). To get the 0.3 bonus, you have to pass at least three exercises, with the fourth one being either completely correct or borderline accepted. The solutions of the exercises are presented the day after the submission deadline respectively. Feel free to use the Moodle forum or visit us during office hours (Tuesday 10:00 – 11:00) to ask questions!

The exercise zip-archive contains Python script for visualization, a Visual Studio 2017 solution with a project folder for each task. The required data is located in the data folder. We provide a pre-compiled version of Ceres on Windows for Visual Studio 2017 (and above). For other environments it can be compiled following instructions on the provided website.

**Expected submission files:** gaussian.cpp, surface.cpp, dragon.cpp, output_gaussian.txt, output_surface.txt, output_dragon.txt, gaussian.png, surface.png, dragon.png

# Exercise 3 – Gaussian – Surface – Registration

In this exercise, we explore different optimization problems, that can be solved with a (non-)linear solver, Ceres (http://ceres-solver.org/) in our case. For each exercise save the console log output and the plot.

# Tasks:

## 0. Setup

We provide scripts to visualize your results. These scripts are implemented in Python 3 (https://www.python.org/) using the library Matplotlib (https://www.matplotlib.org/).

After installing Python 3 run the following command in a terminal to install all dependencies:

```
python -m pip install --user numpy scipy matplotlib
```

Open a terminal in the exercise folder and try to run one of Python scripts to see if your installation was successful.

## 1. Gaussian curve

In the first exercise we want to fit a Gaussian curve to a collection of points drawn from an unknown, noisy Gaussian probability density function:

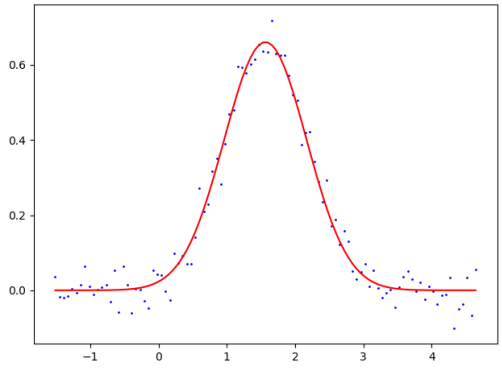
$$f(x) = \frac{1}{\sqrt{2\,\pi\,\sigma^2}}\; e^{-\frac{(x-\mu)^2}{2\,\sigma^2}}$$

Your task is to find the values of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ of the model that best fits the data in `points_gaussian.txt` using Ceres.

Implement the `operator()` function of `GaussianCostFunction` in `gaussian.cpp` and run the program.

Visualize your resulting model by running the `plot_gaussian.py` script with your values for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$:

```
python plot_gaussian.py --mu <value> --sigma <value>
```




## 2. 3D Surface

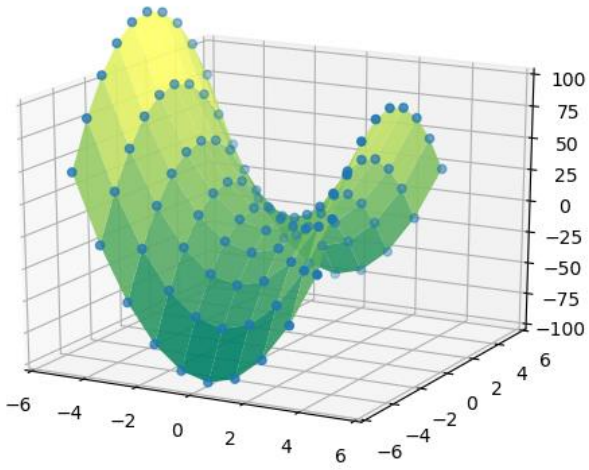
In the second exercise your task is to implement a hyperbolic paraboloid in the form of

$$c * z = \frac{x^2}{a} - \frac{y^2}{b}$$

Similar to Task 1) find the values for $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ that best fits the values of `points_surface.txt`. Finish the implementation in `surface.cpp`: Read in the input data, create a new cost function and define the parameters for the problem.

Use `plot_surface.py` with your parameters to visualize your result:

```
python plot_surface.py --a <value> --b <value> --c <value>
```



## 3. Registration

In this task we look at a simple marker-based registration problem. Given two pointclouds $P$ and $Q$ sparsely sampled from the silhouette of the famous Stanford Dragon model with known 1:1 correspondences, we want to find the 2D transformation $T$ (rotation & translation) that minimizes the error. Additionally, each correspondence comes with a specific weight.

$$error = \sum_i w_i \parallel T\, p_i - q_i \parallel^2$$

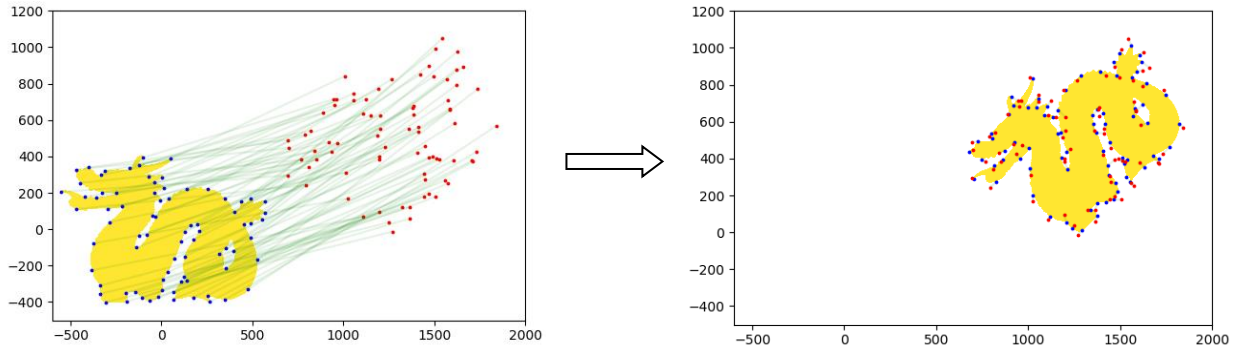
$$T(\theta, t) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$p_i \in P \quad q_i \in Q$$

Your task is to finish the implementation in `dragon.cpp`: Use the data from `points_dragon_1.txt`, `points_dragon_2.txt` and `weights_dragon.txt`.

Visualize your result using `plot_dragon.py` and your values.

```
python plot_dragon.py --deg <value> --tx <value> --ty <value>
```




## 4. Submit your solution

Task 1) Code (gaussian.cpp), console log (output_gaussian.txt), plot (gaussian.png)

Task 2) Code (surface.cpp), console log (output_surface.txt), plot (surface.png)

Task 3) Code (dragon.cpp), console log(output_dragon.txt), plot (dragon.png)