November 9, 2024

General Information:

You have to submit your solution via Moodle. We allow **groups of two students**. Please list all names and matriculation numbers in the **team_members.txt** file. **Both team members need to upload the solution**. You have **one week of working time** (note the deadline date in the footer). To get the 0.3 bonus, you have to pass at least four exercises, with the fifth 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 use a Q&A session to ask questions!

The zip-archive for the exercise contains the source templates and CMake configuration file. Data is provided in the separate zip.

Expected submission files:

- **Code Files:** gaussian.cpp, surface.cpp, dragon.cpp,
- **Console Logs:** output_gaussian.txt, output_surface.txt, output_dragon.txt,
- **Plots:** gaussian.png, surface.png, dragon.png
- (optional) team_members.txt

Exercise 3 – Optimization

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:

1. Setup

1.1. File Structure

```
Exercises/
    Data/
        points_gaussian.txt
        points_surface.txt
        ...
    Libs/
        ...
    Exercise-4/
        Build/
            CMake Files and Binaries
        gaussian.cpp
        ...
```

1.2. Python

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
```

(consider creating a dedicated virtual environment first)

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

1.3. glog

Ceres depends on glog (https://github.com/google/glog), so we first need to build this library.
1) Get the sources:

- Download the distribution from moodle
- Or clone the version 0.5.0

2) Open CMake and configure it. We are not going to use gflags, so **unselect** the entries **“WITH_GFLAGS”** and **“BUILD_TESTING”**. Set the entry **“CMAKE_INSTALL_PREFIX”** to “Libs/Glog” (If you do not see this entry, check the “Advanced” checkbox at the top right.) This path should also not include spaces.

3) Build the library by running the **“glog”** project and install it by building the project **“INSTALL”**. You should now have a folder called “Libs/Glog” which contains an “include” and a “lib” folder.

1.4. Ceres

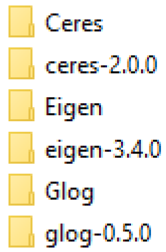
1) Get the sources:

- Download the distribution from moodle
- Or clone the version 2.0.0

2) Run CMake and configure it (you might get error messages). Again, we are not going to use gflags, so unselect the entries **“BUILD_EXAMPLES”**, **“BUILD_BENCHMARKS”**, **“BUILD_TESTING”, “GFLAGS”**. Set the entry **“Eigen3_DIR”** to the Eigen installation folder (e.g. “Libs/Eigen”), set “glog_DIR” to the glog installation folder (e.g. “Libs/Glog”). Also deactivate **LAPACK, CUSTOM_BLAS, SUITESPARSE**

Windows: Set the entry **“CMAKE_INSTALL_PREFIX”** to “Libs/Ceres” (If you do not see this entry, check the “Advanced” checkbox at the top right.) This path should also not include spaces.

3) Build the library, this might take a while and install it (as you did with glog). In the meantime you can browse the Ceres tutorial page (http://ceres-solver.org/nnls_tutorial.html) to get a sense of the type of problems we are going to address.



Example "Libs" folder structure on Windows

1.5. Exercise

1) Download the exercise from moodle and extract it.
2) Configure it with CMake and make sure, it finds Eigen, glog and Ceres
3) For each of the 3 tasks, there is a separate project.

2. Tasks

2.1. Gaussian curve

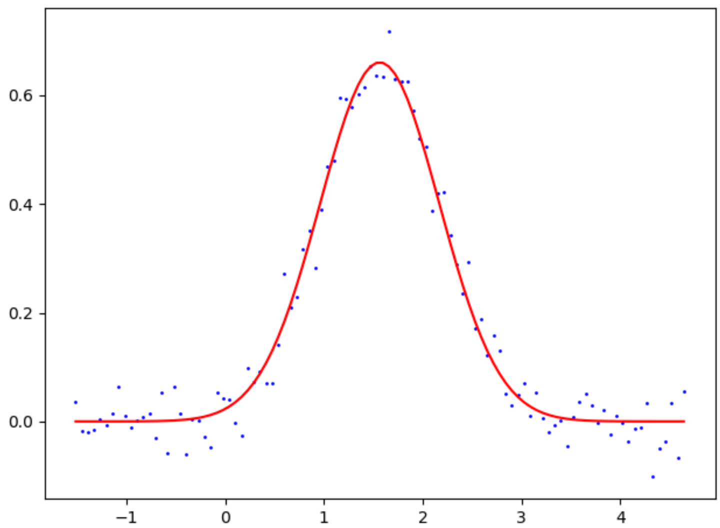
In the first task we want to fit a Gaussian curve to a collection of points drawn from an unknown, noisy Gaussian probability density function:

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

Your task is to use Ceres to find the values of μ and σ of the model that best fits the data in `points_gaussian.txt`.
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 obtained values for μ and σ:

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



2.2. 3D Surface

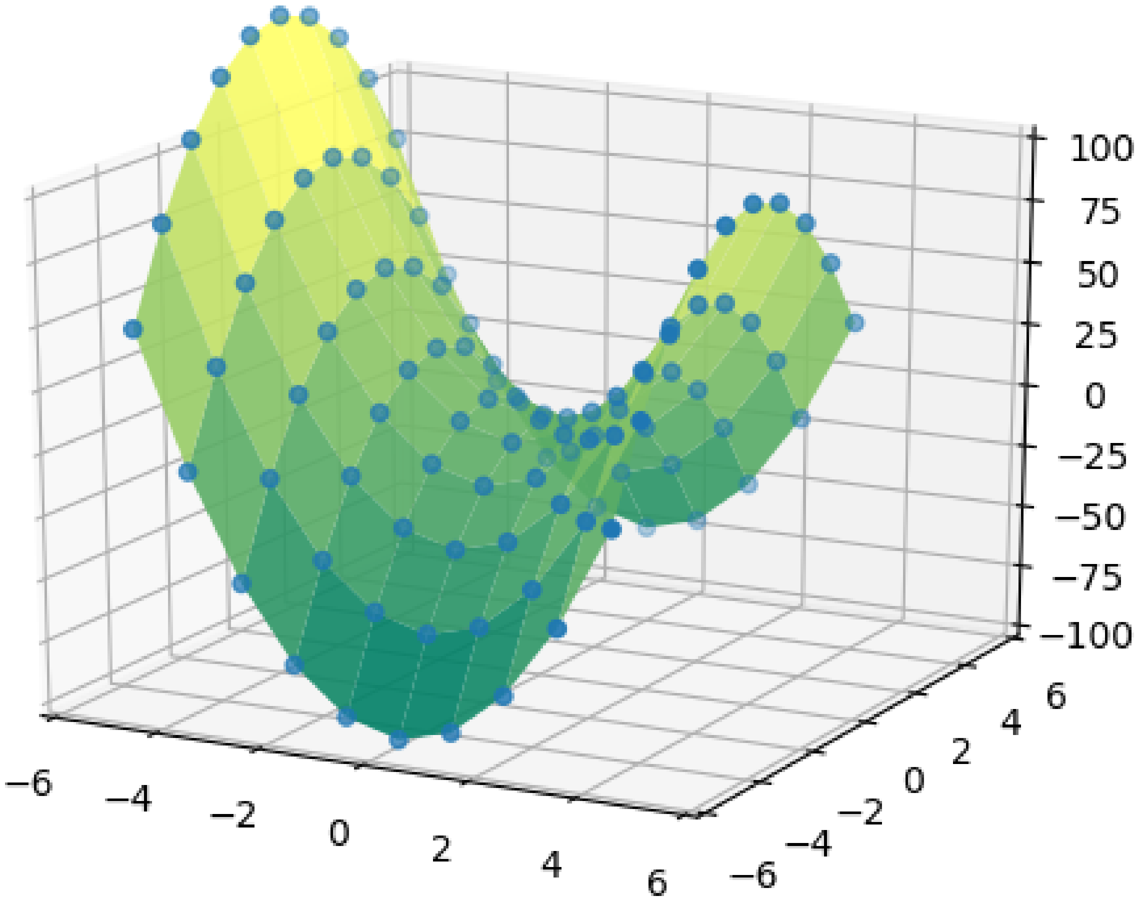
In the second task we want to optimize for a hyperbolic paraboloid in the form of

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

Your task is to Ceres to find the values for a, b and c that best fit 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>
```



2.3. Registration

In this task we look at a simple marker-based registration problem. Given two point clouds 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.

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

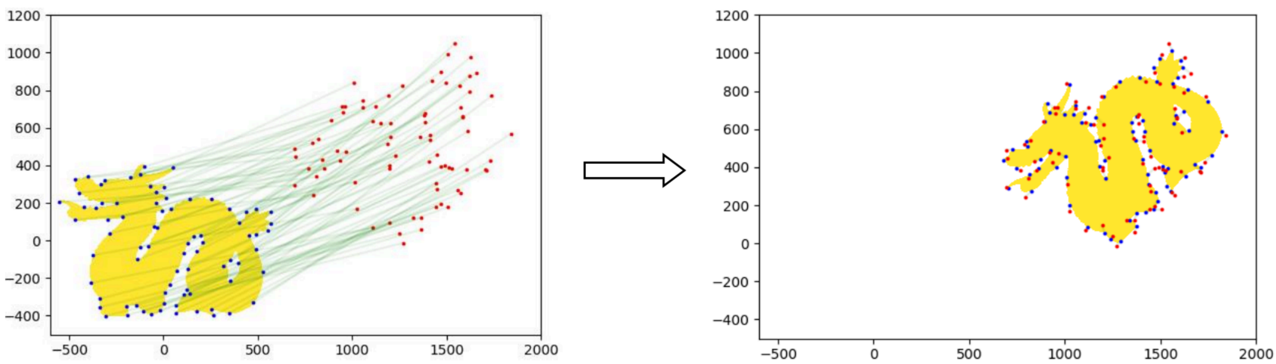
$$T(\theta, t) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

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

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>
```



3. 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**)
- (Optional) If you worked in a group, upload **team_members.txt**.