3D Scanning & Motion Capture

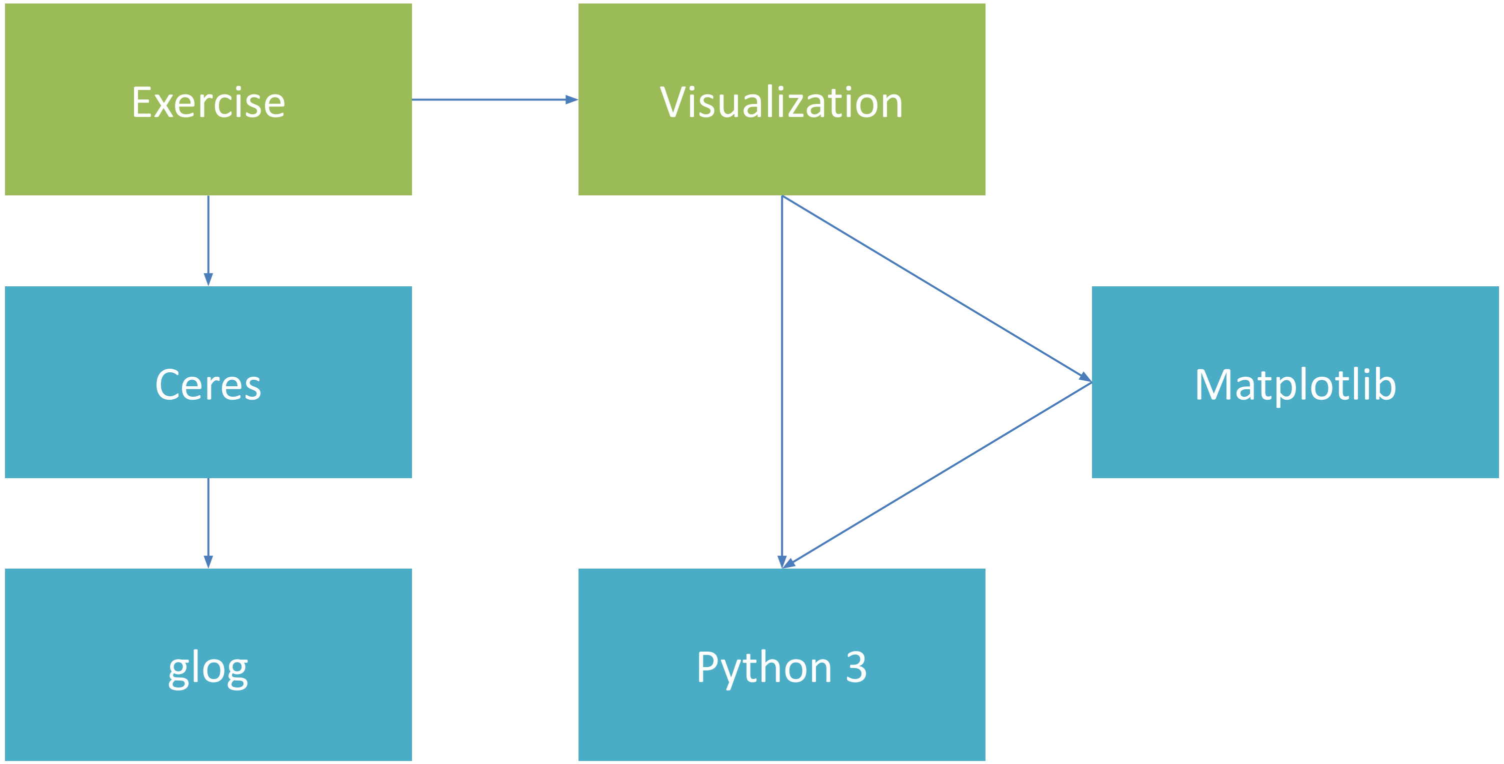
Exercise - 4

Andrei Burov, Yuchen Rao

Exercises – Overview

1. Exercise → Camera Intrinsics, Back-projection, Meshes
2. Exercise → Surface Representations
3. Exercise → Coarse Alignment (Procrustes)
4. **Exercise → Optimization**
5. Exercise → Object Alignment, ICP

Project Dependencies



Project Dependencies – 1) Python

- Download Python 3.+ from https://www.python.org/
- Install python libraries using pip
  - Numpy, Matplotlib, Scipy
- Try if exercise visualization scripts work
- **Note:** use **-h** flag to see python script options
  - e.g. "python plot_gaussian.py -h" ⟶

```
usage: plot_gaussian.py [-h] [--data_path DATA_PATH] [--mu MU] [--sigma SIGMA]

optional arguments:
  -h, --help            show this help message and exit
  --data_path DATA_PATH
                        Directory of the data files.
  --mu MU               The mu value from Ceres.
  --sigma SIGMA         The sigma value from Ceres.
```

Project Dependencies – 2) glog

- Download sources from https://github.com/google/glog.git
- Configure glog with CMake
  - Disable gflags and tests
  - Optional/Windows: Set install path to “Libs” folder
- Build and install

Project Dependencies – 3) Ceres

- Download sources from https://ceres-solver.googlesource.com/ceres-solver
- Configure Ceres with CMake
  - Set path to Eigen install directory
  - Disable gflags, benchmarks, tests and examples, LAPACK
  - Optional/Windows: Set install path to “Libs” folder
- Build and install

Project Configuration – Exercise

- Download exercise sources from moodle
- Configure with CMake
  - Make sure Eigen, glog and Ceres paths are set
- Build

Ceres

- Open source C++ library for modeling and solving non-linear optimization problems

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\sum_i \rho_i\left(\left\|f_i\left(x_{i_1},\ldots,x_{i_k}\right)\right\|^2\right) \\ \text{s.t.} \quad & l_j \le x_j \le u_j \end{aligned}$$

- Special case:

$$\frac{1}{2}\sum_i \left\|f_i\left(x_{i_1},\ldots,x_{i_k}\right)\right\|^2$$

How to Use Ceres

1. Define optimization problem with templated functors

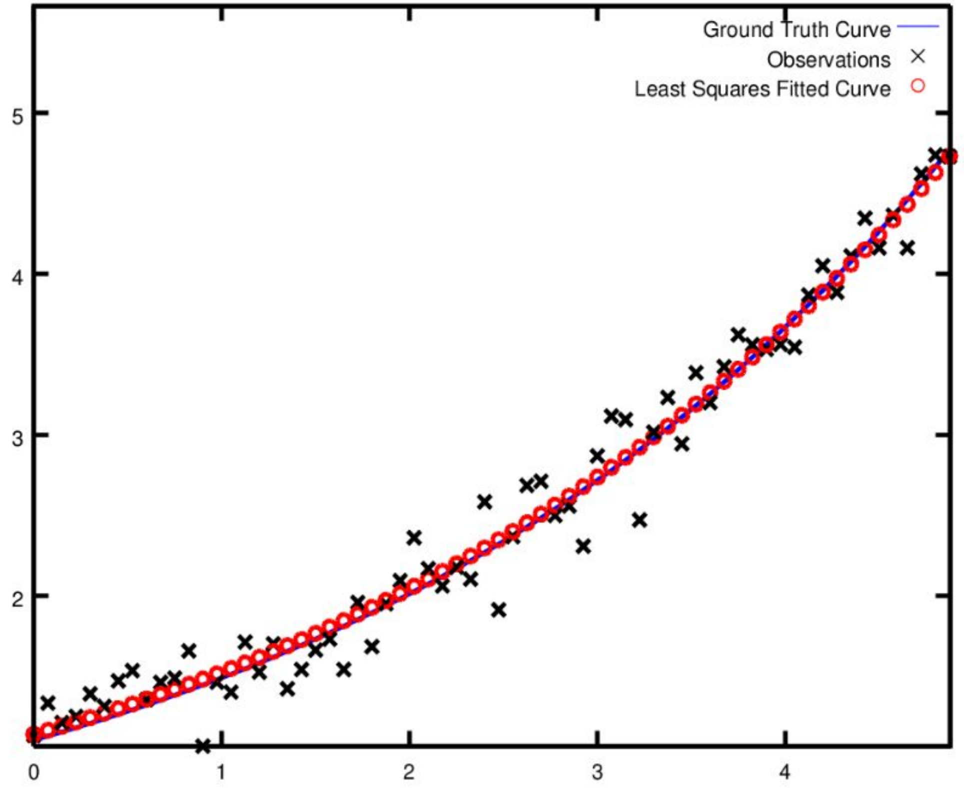
2. Define initial parameter values

3. Add a residual block per functor for each data point

4. Define Ceres options & solve

Detailed Tutorial: http://ceres-solver.org/nnls_tutorial.html

How to Use Ceres - Curve Fitting Example



$$y = e^{mx+c}$$

Full example code:

https://ceres-solver.googlesource.com/ceres-solver/+/master/examples/curve_fitting.cc

How to Use Ceres - Curve Fitting Example

1. Define optimization problem with templated functors

```
struct ExponentialResidual {
  ExponentialResidual(double x, double y) : x_(x), y_(y) {}

  template <typename T>
  bool operator()(const T* const m, const T* const c, T* residual) const {
    residual[0] = y_ - exp(m[0] * x_ + c[0]);
    return true;
  }

 private:
  const double x_;
  const double y_;
};
```

Parameters (what to optimize)

Data (Fixed)

Residuals (what to minimize)
⟶ Ceres automatically squares them!

Full example code:

https://ceres-solver.googlesource.com/ceres-solver/+/master/examples/curve_fitting.cc

How to Use Ceres - Curve Fitting Example

2. Define initial parameter values

```
double m = 0.0;
double c = 0.0;
```

Full example code:

https://ceres-solver.googlesource.com/ceres-solver/+/master/examples/curve_fitting.cc

How to Use Ceres - Curve Fitting Example

3. Add a residual block per functor for each data point

Size of residuals

Size of each parameter

```
Problem problem;
for (int i = 0; i < kNumObservations; ++i) {
  problem.AddResidualBlock(
      new AutoDiffCostFunction<ExponentialResidual, 1, 1, 1>(
          new ExponentialResidual(data[2 * i], data[2 * i + 1])),
      NULL,
      &m,
      &c);
}
```

data

initial parameters

Reference: http://ceres-solver.org/nnls_modeling.html#_CPPv4N5ceres20AutoDiffCostFunctionE

How to Use Ceres - Curve Fitting Example

4. Define Ceres options & Solve

```
Solver::Options options;
options.max_num_iterations = 25;
options.linear_solver_type = ceres::DENSE_QR;
options.minimizer_progress_to_stdout = true;

Solver::Summary summary;
Solve(options, &problem, &summary);
std::cout << summary.BriefReport() << "\n";
std::cout << "Initial m: " << 0.0 << " c: " << 0.0 << "\n";
std::cout << "Final   m: " << m << " c: " << c << "\n";
```

Full example code:

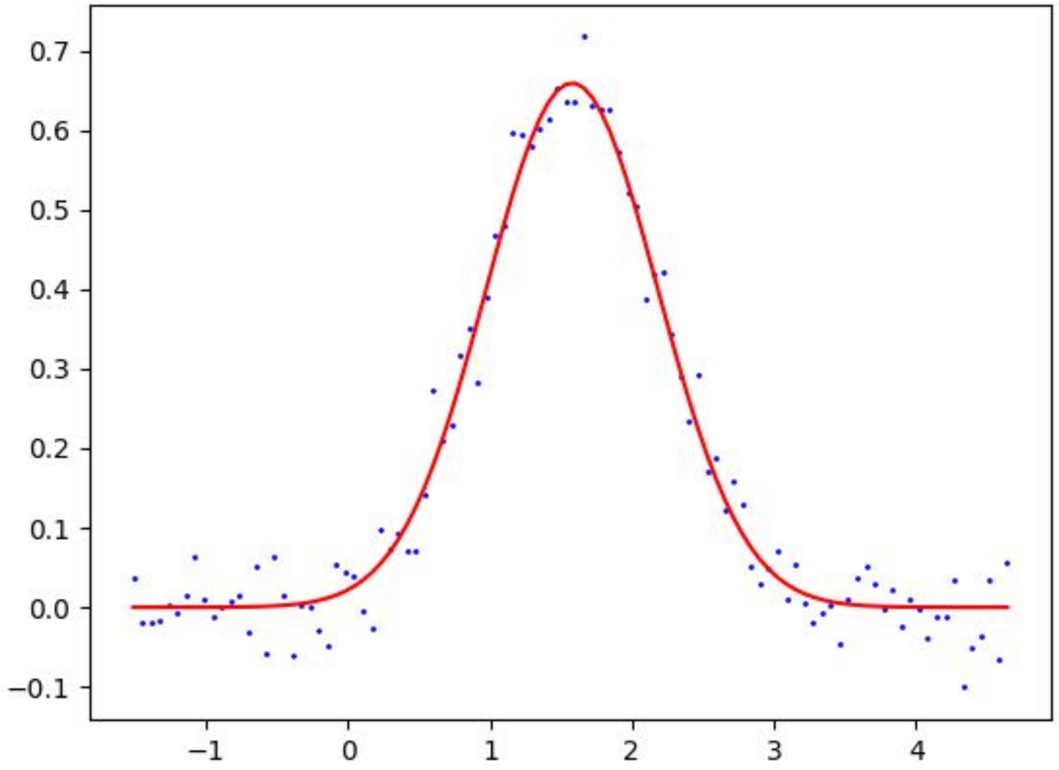
https://ceres-solver.googlesource.com/ceres-solver/+/master/examples/curve_fitting.cc

Task 1) Gaussian

- **Data:** Set of 2D points
  - Drawn from a Gaussian PDF

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

  - Contain noise

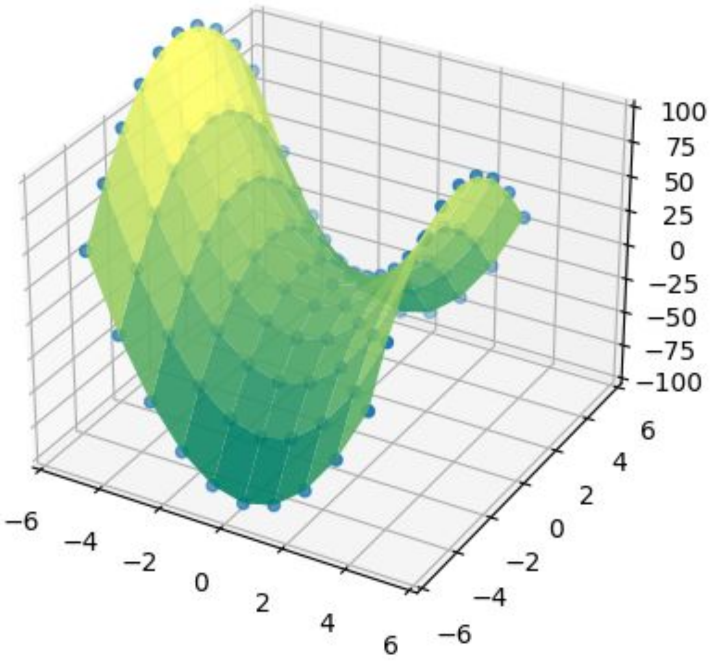


- **Goal:** Find Gaussian parameters
  - µ and σ

Task 2) 3D Surface

- **Data:** Set of 3D points
  - Drawn from a hyperbolic paraboloid

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



- **Goal:** Find surface parameters
  - a, b, and c

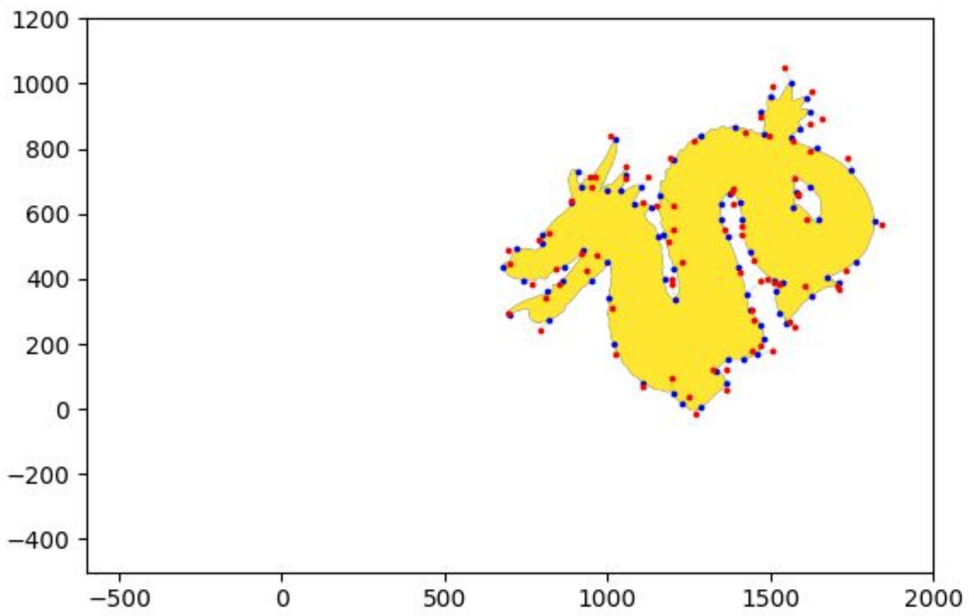
Task 3) Registration

- **Data:** Two Sets of 2D points
  - Source and transformed target

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

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

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



- **Goal:** Find transformation parameters
  - Rotation θ and 2D translation $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$

See you next time!