Computer Vision - FMAN85
Assignment 5 - Spring 2020

Local Optimization and Structure from Motion

Hicham Mohamad, hi8826mo-s
hsmo@kth.se

February 21, 2020

1 Introduction

In this assignment, we are going to deal with **Model Fitting using optimization**. In particular, we will compute the **maximal likelihood** solution for a couple of structure and motion problems.

The resulting plots and values in this report are obtained by running the implemented Matlab scripts `ass5_CE1.m` and `ass3_CE3_CE4.m`.

2 Maximum Likelihood Estimation for Structure from Motion Problems

2.1 Exercise 1

Suppose the 2D-point $x_{ij} = (x_{ij}^1, x_{ij}^2)$ is an observation of the 3D-point $\mathbf{X}_j$ in camera P_i. Also we assume that the observations are corrupted by Gaussian noise, that is,

$$(x_{ij}^1, x_{ij}^2) = \left(\frac{P_i^1 \mathbf{X}_j}{P_i^3 \mathbf{X}_j}, \frac{P_i^2 \mathbf{X}_j}{P_i^3 \mathbf{X}_j} \right) + \epsilon_{ij} \tag{1}$$

where P_i^1, P_i^2, P_i^3 are the rows of the camera matrix P_i and ϵ_{ij} is normally distributed with covariance σI. The **probability density function** is then

$$p(\epsilon_{ij}) = \frac{1}{2\pi\sigma} e^{-\frac{1}{2\sigma^2}||\epsilon_{ij}||^2} \tag{2}$$

Assuming that the ϵ_{ij} are independent, that is

$$p(\epsilon) = \prod_{i,j} p(\epsilon_{ij}), \tag{3}$$

Maximizing the log-likelihood

By using logarithm calculus and plugging in 2 we can write

$$\begin{aligned} \log p(\epsilon) &= \log \left(\prod_{i,j} p(\epsilon_{ij}) \right) = \sum \log \left(p(\epsilon_{ij}) \right) \\ &= \sum_{i,j} \log \left(\frac{1}{2\pi\sigma} \right) - \frac{1}{2\sigma^2} \sum_{ij} || \left((x_{ij}^1 - \frac{P_i^1 \mathbf{X}_j}{P_i^3 \mathbf{X}_j}, x_{ij}^2 - \frac{P_i^2 \mathbf{X}_j}{P_i^3 \mathbf{X}_j} \right) ||^2 \end{aligned} \tag{4}$$

Now we can maximize the log-likelihood and write

$$\begin{aligned}\max \log p(\epsilon) = \max \quad & \sum_{i,j} \log\left(\frac{1}{2\pi\sigma}\right) - \frac{1}{2\sigma^2}\sum_{ij} || \left((x_{ij}^1 - \frac{P_i^1 \mathbf{X}_j}{P_i^3 \mathbf{X}_j}, x_{ij}^2 - \frac{P_i^2 \mathbf{X}_j}{P_i^3 \mathbf{X}_j}\right) ||^2 \\ = \max \quad & -\sum_{ij} || \left((x_{ij}^1 - \frac{P_i^1 \mathbf{X}_j}{P_i^3 \mathbf{X}_j}, x_{ij}^2 - \frac{P_i^2 \mathbf{X}_j}{P_i^3 \mathbf{X}_j}\right) ||^2\end{aligned} \tag{5}$$

by changing the minus sign we get a minimizing problem. In this way, we show that the model configuration (points and cameras) that maximizes the likelihood of the obtaining observations $x_{ij} = (x_{ij}^1, x_{ij}^2)$ is obtained by solving

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m} || \left(x_{ij}^1 - \frac{P_i^1 \mathbf{X}_j}{P_i^3 \mathbf{X}_j}, x_{ij}^2 - \frac{P_i^2 \mathbf{X}_j}{P_i^3 \mathbf{X}_j}\right) ||^2 \tag{6}$$

3 Calibrated Structure from Motion and Local Optimization

3.1 Computer Exercise 1

In Computer Exercise 3 and 4 of Assignment 3, we computed a solution to the two-view structure form motion problem for the two images of a part of the fort Kronan in Gothenburg using the 8-point algorithm. In this exercise the goal is to use the **solution from Assignment 3** as a starting solution and locally improve it using the **Levenberg-Marquardt** method.

For solving the task, it is provided the following useful functions

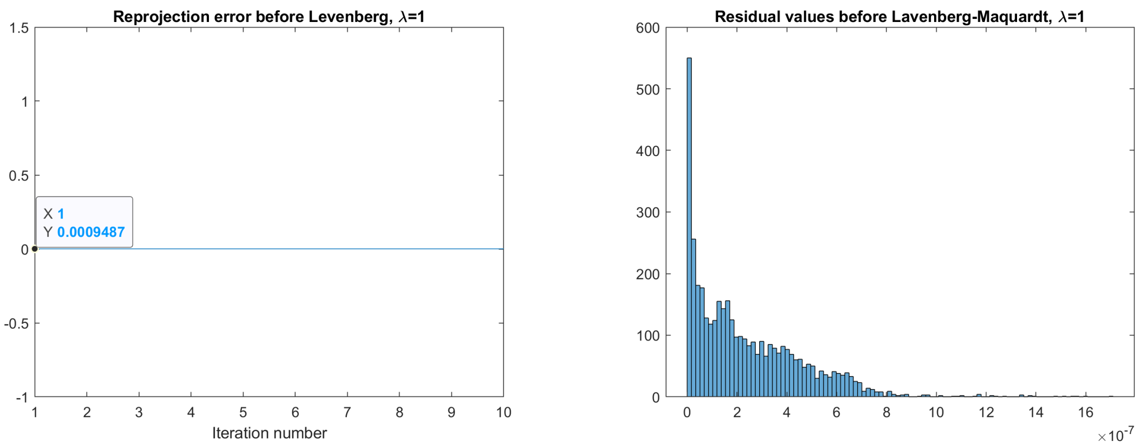
- `LinearizeReprojErr.m` contains a function that for a given set of cameras, 3D points and image points, computes the linearization

$$r(v) \approx r(v_0) + J(v_0)\delta v \tag{7}$$

  where $\delta v = v - v_0$ and $J(v_0)$ is a matrix whose rows are the gradients of $r_i(v)$ at v_0.

- `update_solution.m` contains a function that computes a new set of cameras and 3D points from an update δv computed by any method.

- `ComputeReprojectionError.m` computes the reprojection error for a given set of cameras, 3D points and image points. It also returns the values of all the individual residuals as a second output.



(a) (b)

Figure 1: Plots before running the Levenberg-Maquardt for $\lambda = 1$. a) plot of the reprojection error versus the iteration number. b) Plot of histogram of all the residual values.

3.1.1 Using Levenbergh-Maquardt

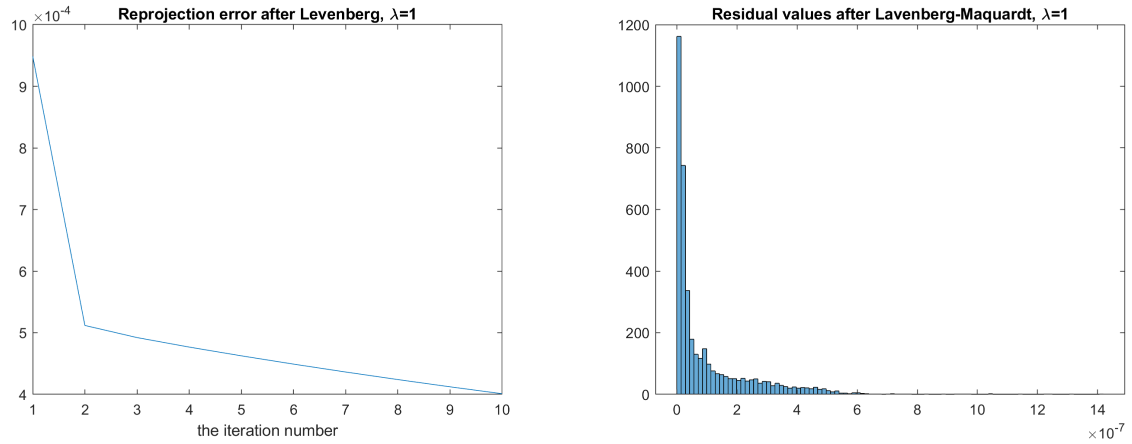
In the Levenberg-Maquardt method the **update** is given by

$$\delta v = -(J(v_k)^T J(v_k) + \lambda I)^{-1} J(v_k)^T r(v_k). \tag{8}$$

The Levenberg-Marquardt update is often much more stable than the original Gauss-Newton formulation. If λ is selected large enough the update $v_1 = v_0 + d$ is guaranteed to give a better objective value. On the other hand for a very large λ the update in 8 is almost the same as

$$\delta v = J(v_k)^T r(v_k) \tag{9}$$

because it becomes multiplied with $1/\lambda$.



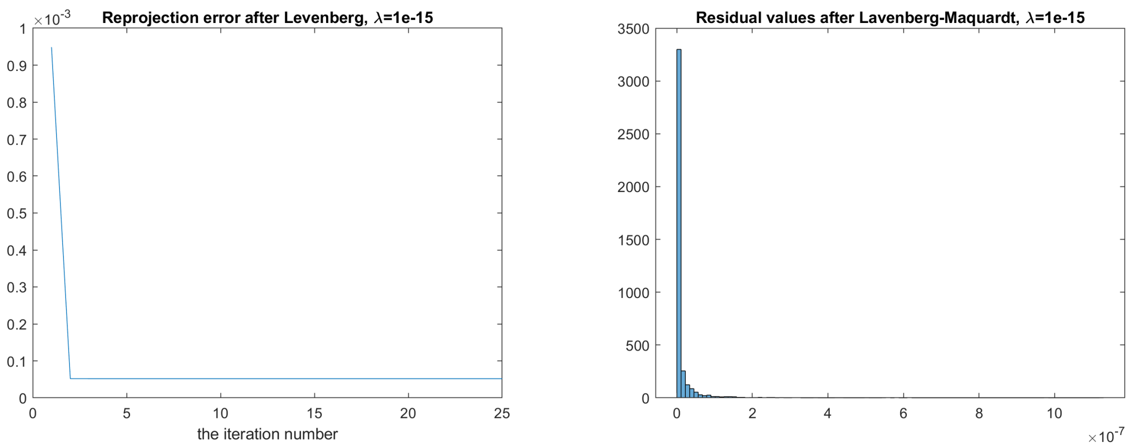
(a) (b)

Figure 2: Plotting after running the Levenberg-Maquardt method, when $\lambda = 1$. a) Plot of the reprojection error. b) Plot of histogram of all the residual values.

A **common strategy** is to start with a large λ and gradually reducing it to increase **convergence speed** when approaching the minimum value.

Using the scheme in 8 and starting from the solution that we got in Assignment 3, we plot the reprojection error versus the iteration number for $\lambda = 1$, as shown in Figure 2a. Also we plot histograms of all the residual values after running the Levenberg-Maquardt method, as shown in Figure 2b.

As mentioned before, by trying small $\lambda = 10^{-15}$ we get faster convergence as illustrated in Figure 3.



(a) (b)

Figure 3: a) plot of the reprojection error versus the iteration number for $\lambda = 10^{-15}$ after running the Levenberg-Maquardt. b) plot of histograms of all the residual values after running the Levenberg-Maquardt method in when $\lambda = 10^{-15}$.

References

[1] Carl Olsson, Computer Vision - FMAN85, Lectures notes: https://canvas.education.lu.se/courses/3379

[2] Hartley, Zisserman, Multiple View Geometry, 2004.

[3] Szeliski, Computer Vision - Algorithms and Applications, Springer.