The sparse Levenberg-Marquardt algorithm for bundle adjustment

Daniel O'Connor

These notes describe the sparse Levenberg-Marquardt algorithm for bundle adjustment. I learned this algorithm from the book Multiple View Geometry in Computer Vision by Hartley and Zisserman.

1 Notation

For notational convenience, a 3×4 camera matrix $\begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix}$ will be represented by the vector $P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \in \mathbb{R}^{12}$. This vector $P \in \mathbb{R}^{12}$ will be called a "camera vector". (Here p_1, p_2 and p_3 are vectors in $\mathbb{R}^4$.)

It will be helpful to define the function $\hat{x}$ which takes as input a camera vector

$$P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \in \mathbb{R}^{12}$$

and a point $X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3$ and returns as output the projection of X using the camera vector P. In detail,

$$\hat{x}\left(\begin{bmatrix} P \\ X \end{bmatrix}\right) = \begin{bmatrix} \frac{p_{11}x+p_{12}y+p_{13}z+p_{14}}{p_{31}x+p_{32}y+p_{33}z+p_{34}} \\ \frac{p_{21}x+p_{22}y+p_{23}z+p_{24}}{p_{31}x+p_{32}y+p_{33}z+p_{34}} \end{bmatrix} \in \mathbb{R}^2.$$

The input to $\hat{x}$ is the concatenation of P and X, which is a vector in $\mathbb{R}^{15}$. To save paper, we will usually write $\hat{x}(P, X)$ rather than the more cumbersome expression $\hat{x}\left(\begin{bmatrix} P \\ X \end{bmatrix}\right)$.

Throughout these notes, we will often use block notation when writing vectors and matrices.

2 Optimization problem

Our goal in bundle adjustment is to find camera vectors $P_i \in \mathbb{R}^{12}$ (for $i = 1, \ldots, n$) and points $X_j \in \mathbb{R}^3$ (for $j = 1, \ldots, m$) such that

$$\hat{x}(P_i, X_j) \approx x_{ij} \quad \text{for all } i, j.$$

The points $x_{ij} \in \mathbb{R}^2$ are given, and the vectors P_i and X_j are unknown (but we have initial guesses for them). To be precise, our goal is to solve the optimization problem

$$\text{minimize} \quad \sum_{i,j} \frac{1}{2} \|\hat{x}(P_i, X_j) - x_{ij}\|_2^2. \tag{1}$$

The optimization variables are $P_1, \ldots, P_n \in \mathbb{R}^{12}$ and $X_1, \ldots, X_m \in \mathbb{R}^3$.

3 Levenberg-Marquardt algorithm

Problem (1) is a nonlinear least squares problem. However, we could linearize the nonlinear function $\hat{x}$ using the first-order approximation

$$\hat{x}(P + \Delta P, X + \Delta X) \approx \hat{x}(P, X) + \underset{\substack{\uparrow \\ 2\times 12}}{\frac{\partial \hat{x}(P, X)}{\partial P}} \underset{\substack{\uparrow \\ 12\times 1}}{\Delta P} + \underset{\substack{\uparrow \\ 2\times 3}}{\frac{\partial \hat{x}(P, X)}{\partial X}} \underset{\substack{\uparrow \\ 3\times 1}}{\Delta X} \tag{2}$$

to obtain a linear least squares problem. The Gauss-Newton method solves a new instance of this linear least squares problem at each iteration (always linearizing about the most recent estimates of the vectors P_i and X_j). The Levenberg-Marquardt algorithm adds ℓ_2-regularization to the linear least squares problems that are solved at each iteration. In other words, the Levenberg-Marquardt iteration is

$$P_i^{k+1} = P_i^k + \Delta P_i^k, \quad X_j^{k+1} = X_j^k + \Delta X_j^k \quad \text{for } i = 1, \ldots, n, j = 1, \ldots m$$

where the vectors ΔP_i^k and ΔX_j^k are obtained by solving the optimization problem

$$\begin{aligned} \text{minimize} \quad L = \sum_{i,j} \frac{1}{2} \|\hat{x}(P_i^k, X_j^k) + \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P} \Delta P_i + \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X} \Delta X_j - x_{ij}\|_2^2 \\ + \frac{\lambda^k}{2} \sum_i \|\Delta P_i\|_2^2 + \frac{\lambda^k}{2} \sum_j \|\Delta X_j\|_2^2. \end{aligned}$$

The optimization variables (in other words, the inputs to the function L) are the vectors $\Delta P_i \in \mathbb{R}^{12}$ and $\Delta X_j \in \mathbb{R}^3$ (for $i = 1, \ldots, n$ and $j = 1, \ldots, m$). The regularization terms

encourage the vectors ΔP_i and ΔX_j to be small, which means that the approximation (2) is accurate.

The scalar λ^k is chosen adaptively, as follows. If

$$\sum_{i,j} \frac{1}{2} \|\hat{x}(P_i^{k+1}, X_j^{k+1}) - x_{ij}\|_2^2 < \sum_{i,j} \frac{1}{2} \|\hat{x}(P_i^k, X_j^k) - x_{ij}\|_2^2$$

(so the objective function value is reduced), then the vectors P_i^{k+1}, X_j^{k+1} are *accepted* and $\lambda^{k+1} = \lambda^k/10$. Otherwise, the vectors P_i^{k+1}, X_j^{k+1} are *rejected*. In this case, we increase the value of λ^k by a factor of 10, and recompute the vectors P_i^{k+1}, X_j^{k+1}. This process is continued until the vectors P_i^{k+1}, X_j^{k+1} are accepted (in other words, until a reduction in the objective function value is achieved).

4 Solving the Levenberg-Marquardt least squares problems

To solve a linear least squares problem, we need only set the gradient of the objective function L equal to 0 and solve the resulting linear system of equations. Setting the derivative of L with respect to ΔP_i equal to 0, then taking the transpose of both sides, we obtain

$$\begin{aligned}
&\sum_j \left(\hat{x}(P_i^k, X_j^k) + \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P} \Delta P_i + \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X} \Delta X_j - x_{ij} \right)^T \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P} + \lambda^k \Delta P_i^T = 0 \\
\implies &\left(\lambda^k I_{12\times 12} + \sum_j \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P}^T \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P} \right) \Delta P_i + \sum_j \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P}^T \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X} \Delta X_j \\
&= \sum_j \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P}^T \left(x_{ij} - \hat{x}(P_i^k, X_j^k) \right).
\end{aligned}$$

Here $I_{12\times 12}$ is the 12×12 identity matrix. Setting the derivative of L with respect to ΔX_j equal to 0, then taking the transpose of both sides, we obtain

$$\begin{aligned}
&\sum_i \left(\hat{x}(P_i^k, X_j^k) + \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P} \Delta P_i + \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X} \Delta X_j - x_{ij} \right)^T \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X} + \lambda^k \Delta X_j^T = 0 \\
\implies &\sum_i \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X}^T \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P} \Delta P_i + \left(\lambda^k I_{3\times 3} + \sum_i \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X}^T \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X} \right) \Delta X_j \\
&= \sum_i \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X}^T \left(x_{ij} - \hat{x}(P_i^k, X_j^k) \right).
\end{aligned}$$

The above equations (for $i = 1, \ldots, n$ and $j = 1, \ldots, m$) can be combined into one single equation as follows:

$$\begin{bmatrix} U_1 & & & & W_{11} & \cdots & W_{1m} \\ & U_2 & & & W_{21} & \cdots & W_{2m} \\ & & \ddots & & \vdots & & \vdots \\ & & & U_n & W_{n1} & \cdots & W_{nm} \\ W_{11}^T & W_{21}^T & \cdots & W_{n1}^T & V_1 & & \\ \vdots & \vdots & & \vdots & & \ddots & \\ W_{1m}^T & W_{2m}^T & \cdots & W_{nm}^T & & & V_m \end{bmatrix} \begin{bmatrix} \Delta P_1 \\ \vdots \\ \Delta P_n \\ \Delta X_1 \\ \vdots \\ \Delta X_m \end{bmatrix} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \\ r_1 \\ \vdots \\ r_m \end{bmatrix} \tag{3}$$

where

$$\begin{aligned} U_i &= \lambda^k I_{12\times 12} + \sum_j \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P}^T \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P}, \\ V_j &= \lambda^k I_{3\times 3} + \sum_i \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X}^T \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X}, \\ W_{ij} &= \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P}^T \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X}, \\ q_i &= \sum_j \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial P}^T \left(x_{ij} - \hat{x}(P_i^k, X_j^k)\right), \\ r_j &= \sum_i \frac{\partial \hat{x}(P_i^k, X_j^k)}{\partial X}^T \left(x_{ij} - \hat{x}(P_i^k, X_j^k)\right) \end{aligned}$$

for $i = 1, \ldots, n, j = 1, \ldots, m$. Equation (3) can also be written more concisely as

$$\begin{bmatrix} U & W \\ W^T & V \end{bmatrix} \begin{bmatrix} \Delta P \\ \Delta X \end{bmatrix} = \begin{bmatrix} q \\ r \end{bmatrix} \tag{4}$$

where

$$U = \begin{bmatrix} U_1 & & \\ & \ddots & \\ & & U_n \end{bmatrix}, V = \begin{bmatrix} V_1 & & \\ & \ddots & \\ & & V_m \end{bmatrix}, W = \begin{bmatrix} W_{11} & \cdots & W_{1m} \\ \vdots & \ddots & \vdots \\ W_{n1} & \cdots & W_{nm} \end{bmatrix}, q = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix}, r = \begin{bmatrix} r_1 \\ \vdots \\ r_m \end{bmatrix}.$$

To solve (4), we first isolate ΔX:

$$\begin{aligned} & W^T \Delta P + V \Delta X = r \\ \implies & \Delta X = V^{-1}(r - W^T \Delta P). \end{aligned} \tag{5}$$

Now plugging this expression into the equation $U\Delta P + W\Delta X = q$, we obtain

$$\begin{aligned} & U\Delta P + WV^{-1}(r - W^T\Delta P) = q \\ \implies & \underbrace{(U - WV^{-1}W^T)}_{12m \times 12m}\Delta P = q - WV^{-1}r. \end{aligned}$$

We can solve this linear system to find ΔP, then plug the result into (5) to find ΔX.

Notice this **key point**: multiplying by V^{-1} is inexpensive because V is block diagonal (with 3×3 blocks). When n is large, this approach to solving (3) is far more efficient than solving (3) naively, without exploiting sparsity.