Chapter 7: Bundle Adjustment and Nonlinear Optimization

Konrad Koniarski

2016/07/25

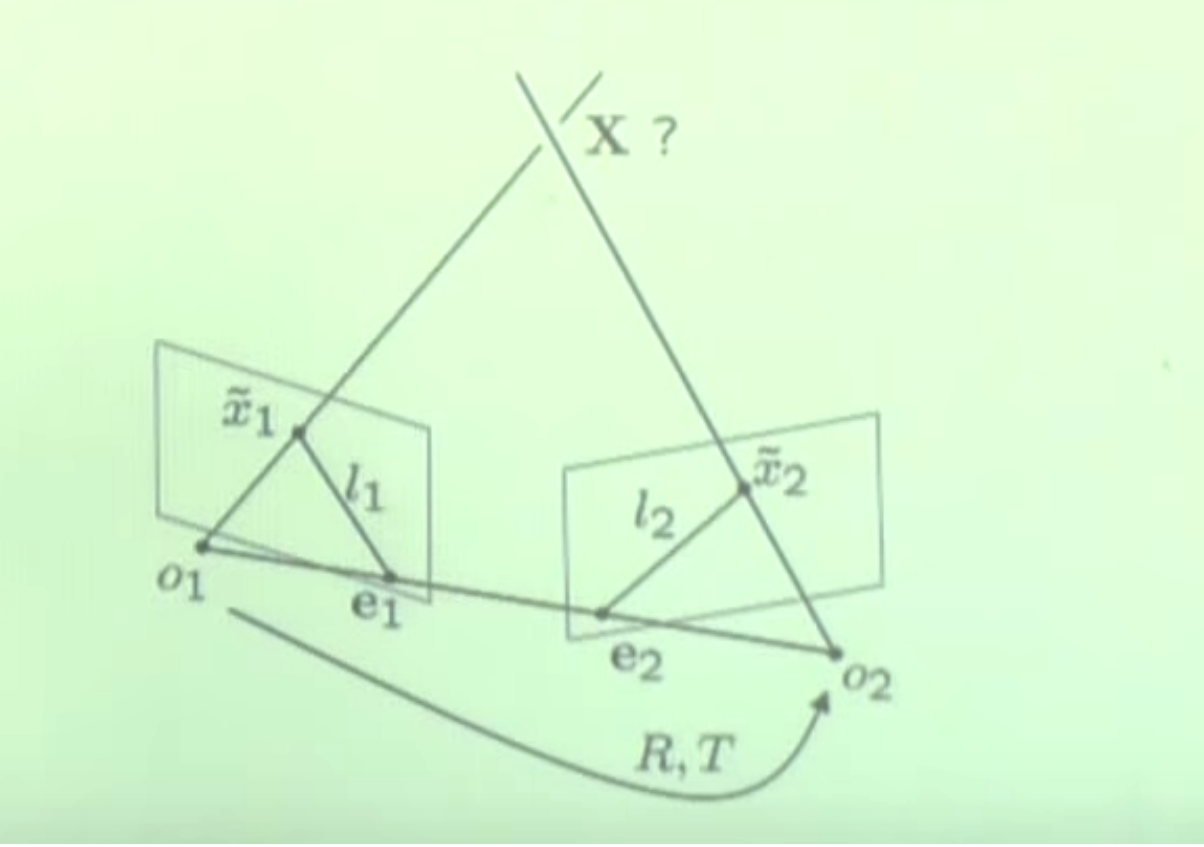
1 Optimality in Noisy Real World Conditions

1.1 Optimality in Noisy Real World Conditions

In the previous chapters we discussed linear approaches to solve the structure and motion problem. In particular, the eight-point algorithm provides closed-form solution to estimate the camera parameters and the 3D structure, based on singular value decomposition.

However, if we have noisy data $\tilde{x}_1$, $\tilde{x}_2$ (correspondences not exact or even incorrect), then we have no guarantee

- that R and T are as close as possible to the true solution.
- that we will get a consistent reconstruction.



1.2 Statistical Approaches to Cope with Noise

The linear approaches are **elegant** because optimal solutions to respective problems can be computed in **closed form**. However, they often fail when dealing with noisy and imprecise point locations. Since measurement noise is not explicitly considered or modeled, such spactral methods often provide **suboptimal performance in noisy real-world conditions**.

In order to take noise and statistical cluctuation into account, one can revert to a **Bayesian formulation** and determine the most likely camera transformation R, T and 'true' 2D coordinates x given the measured coordinates $\tilde{x}$, by performing a **maximum aposteriori estimate**:

$$arg \max_{x,R,T} \mathcal{P}(x, R, T|\tilde{x}) = arg \max_{x,R,T} \mathcal{P}(\tilde{x}|x, R, T)\mathcal{P}(x, R, T) \tag{1}$$

This approach will however involve modeling probability densities $\mathcal{P}$ on the fairly complicated space $SO(3) \times \mathcal{S}^2$ of rotation and translation parameters, as $R \in SO(3)$ and $T \in \mathcal{S}^2$ (3D translation with unit length)

2 Bundle Adjustment

2.1 Bundle Adjustment and Nonlinear Optimization

Under the assumption that the observed 2D point coordinates $\hat{x}$ are corrupted by **zero-mean Gaussian noise**, maximum likehood estimation leads to **bundle adjustment**:

$$E(R, T, X_1, ..., X_N) = \sum_{j=1}^{N} |\tilde{x}_1^j - \pi(X_j)|^2 + |\tilde{x}_2^j - \pi(R, T, X_j)|^2 \tag{2}$$

It aims at minimizing the **reprojection error** between the observed 2D coordinates $\hat{x}_i^j$ and the projected 3D coordinate X_j. Here $\Pi(R, T, X_j)$ denotes the perspective projection X_j after rotation and translation.

For the general case of m **images**, we get:

$$E(\{R_i, T_i\}_{i=1,...,m}, \{X_j\}_{j=1,...,N}) = \sum_{i=1}^{m} \sum_{j=1}^{N} \theta_{ij} |\tilde{x}_i^j - \pi(R_i, T_i, X_j)|^2 \tag{3}$$

with $T_1 = 0$ and $R_1 = 1$. $\theta_{ij} = 1$ **if point** j **is visible in image** i, $\theta_{ij} = 0$ else. The above problems are **non-convex**.

2.2 Different Parameterizations of the Problem

The same optimization problem can be parameterized differently. For example, we can introduce x_i^j to denote the true **2D coordinate** associated with the measured coordinate $\hat{x}_i^j$:

$$E(\{x_1^j, \lambda_1^j\}_{j=1,...,N}, R, T) = \sum_{j=1}^{N} ||x_1^j - \hat{x}_1^j||^2 + ||x_2^j - \pi(R\lambda_1^j x_1 + T)||^2 \tag{4}$$

Alternatively, we can perform a **constrained optimization** by minimizing a cost function (similarity to measurements):

$$E(\{x_i^j\}_{j=1,...,N}.R,T) = \sum_{j=1}^{N}\sum_{i=1}^{2} ||x_i^j - \tilde{x}_i^j||^2 \tag{5}$$

subject to (consistent geometry):

$$x_2^{jT}\hat{T}Rx_1^j = 0, x_1^{jT}e_3 = 1, x_2^{jT}e_3 = 1, j = 1, ..., N \tag{6}$$

2.3 Some Comments on Bundle Adjustment

Bundle adjustment aims at jointly estimating 3D coordinates of points and camera parameters - typically the rigid body motion, but sometimes also intrinsic calibration parameters or radial distortion. Different models of the noise in the observed 2D points leads to different cost functions, zero-mean Gaussian noise being the most common assumption.

The approach is called **bundle adjustment** because it aims at adjusting the bundles of lighrt rays emitted from the 3D points. Originally derived in the field of photogrammetry in the 1950s, it is now used frequently in computer vision. A good overview can be found in **Triggs, McLauchlan, Hartley, Fitzgibbon, "Bundle Adjustment - A Modern Synthesis", ICCV Workshop 1999**.

Typically it is used as a **last step in reconstruction pipeline** because the minimization of this highly non-convex cost function requires a good initialization. The minimization of **non-convex energies** is a challenging problem. Bundle adjustment type cost functions are typically minimized by **nonlinear least squares algorithms**.

3 Nonlinear Optimization

3.1 Nonlinear Programming

Nonlinear programming denotes the process of iteratively solving a nonlinear optimization problem, i.e. a problem involving the maximization or minimization of an objective functional over a set of real variables under a set of equality or inequality constraints.

There are numerous methods and techniques. Good overview of respective methods can be found for example in **Bersekas (1999) "Nonlinear Programming", Nocedal and Wright (1999), "Numerical Optimization" or Luenberg and ye (2008) "Linear and nonlinear programming"**.

Depending on the cost function, different algorithms are employed. In the following, we will discuss **(nonlinear) least squares estimation** and several popular **iterative techniques for nonlinear optimization**:

- the gradient descent
- Newton methods
- the Gauss-Newton algorithm
- the levenberg-Marquardt algorithm.

4 Gradient Descent

4.1 Gradient Descent

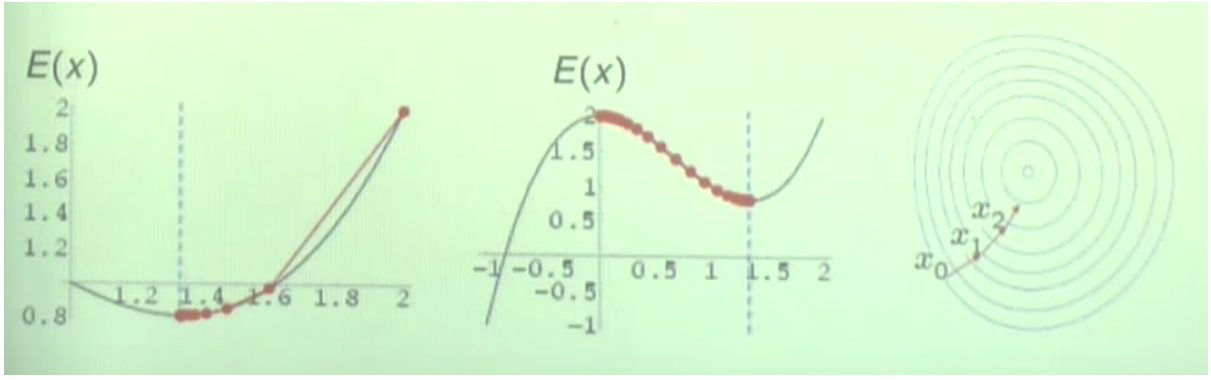
Gradient descent or steepest descent is a first-order optimization method. It aims at computing a local minimum of a (generally) non-convex cost function by iteratively stepping in the direction in which the

energy decreases most. This is given by the negative energy gradient.

To minimize a real-valued cost $E : \mathbb{R}^n \to R$, the gradient flow for $E(x)$ is defined by the differential equation:

$$\begin{cases} x(0) = x_0 \\ \frac{dx}{dt} = -\frac{dE}{dx}(x) \end{cases} \tag{7}$$

Discretization: $x_k + 1 = x_k - \epsilon \frac{dE}{dx}(x_k), k = 0, 1, 2, ...$



4.2 Gradient Descent

Under certain conditions on E(x), the **gradient descent iteration**

$$x_k + 1 = x_k - \epsilon \frac{dE}{dx}(x_k), k = 0, 1, 2, ... \tag{8}$$

converges to a **local minimum**. For the case of **convex** E, this will also be the **global minimum**. The **step size** ϵ can be chosen differently in each iteration.

Gradient descent is a **popular and broadly applicable method**. It is typically not the fastest solution to compute minimizers because the **asymptotic convergence rate is often inferior** to that of more specialized algorithms. First-order methods with optimal convergence rates were pioneered by **Yuri Nesterov**.

In particular, highly anisotropic cost functions (with strongly different curvatures in different directions) require **many iterations** and trajectories tends to zig-zag. Locally optimal step sizes in each iteration can be computed by **line search**. For specific cost functions, alternative techniques such as the **conjugate gradient methods**, **Newton methods**, or the **BFGS method** are prefered.

5 Least Squares Estimation

5.1 Linear Least Squares Estimation

Ordinary least squares or **linear least square** is a method for estimating a set of parameters $x \in \mathbb{R}^d$ in a linear regression model. Assume for each input vector $b_i \in \mathbb{R}, i \in 1, ..., n$, we observe a scalar response $a_i \in \mathbb{R}$. Assume there is a linear relationship of the form

$$a_i = b_i^T x + \eta_i \tag{9}$$

with an unknown vector $x \in \mathbb{R}^d$ and zero-mean Gaussian noise $\eta \sim \mathcal{N}(0, \Sigma)$ with a diagonal covariance matrix of the form $\Sigma = \sigma^2 I_n$. **Maximum likehood estimation** of a x leads to the **ordinary least square** problem:

$$\min_x \sum_i (a_i - x^T b_i)^2 = (a - Bx)^T (a - Bx) \tag{10}$$

Linear least squares estimation was introduced by **Legendre (1805)** and **Gauss (1795/1809)**. When asking for which noise distribution the optimal estimatior was the arithmetic mean, Gauss invented the **normal distribution**.

5.2 Linear Least Squares Estimation

For general Σ, we get the **generalized least squares** problem:

$$\min_x (x - Bx)^T \Sigma^{-1} (a - Bx) \tag{11}$$

This is a quadratic sot function with positive definite Σ_{-1}. It has the **closed-form solution**:

$$\hat{x} = arg\min_x (a - Bx)^T \Sigma^{-1} (a - Bx) = (B^T \Sigma^{-1} B)^{-1} B^T \Sigma^{-1} a \tag{12}$$

If there is no correlation among the observed variances, then the matrix Σ is diagonal. This case is referred to as **weighted least squares**:

$$\min_x \Sigma_i w_i (a_i - x^T b_i)^2, \text{ with } w_i = \sigma_i^{-2} \tag{13}$$

For the case of unknown matrix Σ, there exist iterative estimation algorithms such as **feasible generalization least squares** or **itertively reweighted least squares**.

5.3 Iteratively Reweighted Least Squares

The method of **iteratively reweighted least squares** aims at minimizing generally non-convex optimization problem of the form

$$\min_x \sum_i w_i(x) |a_i - f_i(x)|^2 \tag{14}$$

with some **known weighting function** $w_i(x)$. A solution is obtained by iterating the following problem:

$$x_{t+1} = arg\min_x \sum_i w_i(x_t) |a_i - f_i(x)|^2 \tag{15}$$

For the case that f_i is linear, i.e. $f_i(x) = x^T b_i$, each subproblem

$$x_{t+1} = arg\min_{t+1} \sum_i w_i(x^t) |a_i - x^T b_i(x)|^2 \tag{16}$$

is simply a **weighted least squares problem** that can be solved in closed form. Nevertheless, this iterative approach will generally not converge to a global minimum of the original (nonconvex) problem.

5.4 Nonlinear Least Squares Estimation

Nonlinear least squares estimation aims at fitting observations (a_i, b_i) with a nonlinear model of the form $a_i \approx f(b_i, x)$ for some functions f parameterized with an unknown vector $x \in \mathbb{R}^d$. Minimizing the sum of squares error

$$\min_x \sum_i r_i(x)^2, \text{ with } r_i(x) = a_i - f(b_i, x) \tag{17}$$

is generally a **non-convex optimization problem**.

the optimality condition is given by

$$\sum_i r_i \frac{\partial r_i}{\partial x_j} = 0, \forall j \in \{1, ..., d\} \tag{18}$$

Typically one cannot directly solve these equation. Yet, there exist iterative algorithms for compting approximate solutions, indluding **Newton methods**, the **Gauss-Newton algorithm** and the **Levenberg-Marquardt algorithm**.

6 Newton Methods

6.1 Newton Methods for Optimization

Newton methods are **second order methods**: In contrast to first-order methods like gradient descent, they also make use of second derivatives. Geometrically, Newton methods iteratively approximage the cost function E(x) quadratically and takes a step to the minimizer of this approximation.

Let x_t be the estimated solution after t iterations. Then the Taylor approximation of $E(x)$ in vicinity of this estimate is:

$$E(x) \approx E(x_t) + g^T(x - x_t) + \frac{1}{2}(x - x_t)^T H(x - x_t) \tag{19}$$

The first and second derivative are denoted by the **Jacobian** $g = dE/dx(x_t)$ and the **Hessian matrix** $d^2E/dx^2(x_t)$. For this second-order apprximation, the optimality condition is :

$$\frac{dE}{dx} = g + H(x - x_n) = 0 \tag{20}$$

Setting the next iterative to the minimizer x leads to

$$x_{t+1} = x_t - H^{-1}g \tag{21}$$

6.2 Newton Methods for Optimization

In practice, one often choses a more conservative step size $\gamma \in (0, 1)$:

$$x_{t+1} = x_t - H^{-1}g \tag{22}$$

When applicable, second-order methods are often faster than first-order methods, at least when measured in number of iterations. In particular, there exists a local neighborhood around each optimum where the Newton method converges quadratically for $\gamma = 1$ (if the Hessian is invertible and Lipschitz continuous).

For **large optimization problem**, computing and inverting the Hessian may be challenging. Moreover, since this problem is often not parallelizable, some second order methods do not profit from GPU acceleration. In such cases, one can aim to **iteratively solve the extremality condition (20)**.

In case then H is not positive definite, there exist **quasi-Newton methods** which aim at approximating H or H^{-1} with a positive definite matrix.

7 The Gauss-Newton Algorithm

7.1 The Gauss-Newton Algorithm

The Gauss-Newton algorithm is a method **to solve non-linear least-squares problem** of the form:

$$\min_x \sum_i r_i(x)^2 \tag{23}$$

It can be derived as **an approximation to the Newton method**. The latter iterates:

$$x_{x+1} = x_t - H^{-1}g \tag{24}$$

with the gradient g

$$g_j = 2\sum_i r_i \frac{\partial r_i}{\partial x_j} \tag{25}$$

and the Hessian H:

$$H_{jk} = 2\sum_i (\frac{\partial r_i}{x_j}\frac{\partial r_i}{\partial x_k} + r_i \frac{\partial^2 r_i}{\partial x_j \partial x_k}) \tag{26}$$

Dropping the second order term leads to the **approximation**:

$$H_{jk} \approx 2\sum_i J_{ij}J_{ik}, \text{ with } J_{ij} = \frac{\partial r_i}{\partial x_j} \tag{27}$$

7.2 The Gauss-Newton Algorithm

The approximation

$$H \approx 2J^T J, \text{with the Jacobian } J = \frac{dr}{dx} \tag{28}$$

together with $g = J^T r$, leads to the Gauss-Newton algorithm:

$$x_{x+1} = x_t + \Delta, \text{ with } \Delta = -(J^T J)^{-1} J^T r \tag{29}$$

7.3 The Gauss-Newton Algorithm

The approximation

$$H \approx 2J^T J, \text{ with the Jacobian } J = \frac{dr}{dx} \tag{30}$$

together with $g = J^T r$, leads to the **Gauss-Newton algorithm**:

$$x_{t+1} = x_t + \Delta, \text{ with } \Delta = -(J^T J)^{-1} J^T r \tag{31}$$

In contrast to the Newton algorithm, the Gauss-Newton algorithm **does not require the computation of second derivatives**. Moreover, the above approximation of the Hessian is by construction **positive define**.

This approximation of the Hessian is valid if

$$|\frac{\partial^2 r_i}{\partial x_j \partial x_k}| \ll |\frac{\partial r_i}{\partial x_j}\frac{\partial r_i}{\partial x_k}| \tag{32}$$

This is the case if the **residuum** r_j **is small** or if it is **close to linear** (in which case the second derivatives are small).

8 The Levenberg-Marquard Algorithm

8.1 The Levenberg-Marquardt Algorithm

The Newton algorithm

$$x_{x+1} = x_t - H^{-1} g \tag{33}$$

can be modified (damped):

$$x_{t+1} = x_t - (H + \lambda I_n)^{-1} g \tag{34}$$

to create a hybrid between the **Newton method** ($\lambda = 0$) and a **gradient descent** with step sieze $1/\lambda$ (for $\lambda \to \infty$).

In the same manner, **Levenberg (1944)** suggest to **damp the Gauss-Newton algorithm** for non-linear least squares:

$$x_{t+1} = x_t + \Delta, \text{ with } \Delta = -(J^T J + \lambda I_n)^{-1} J^T r \tag{35}$$

Marquardt (1963) suggested a more **adaptive component-wise damping** of the form:

$$\Delta = -(J^T J + \lambda diag(J^T J))^{-1} J^T r \tag{36}$$

which avoids slow convergence in directions of small gradient.

9 Summary

9.1 Summary

Bundle adjustment was pioneered in the 1950s as a technique for structure and motion estimation in noisy real-world conditions. It aims at estimating the locations of N 3D points X_j and camera motions (R_j, T_j), given noisy 2D projections $\hat{x}_i^j$ in M images. The assumption of zero-mean Gaussian noise on the 2D observations leads to the **weighted nonlinear least squares problem**:

$$E(\{R_i, T_i\}_{i=1..m}, \{X_j\}_{j=1..N}) = \sum_{i=1}^{m} \sum_{j=1}^{N} \theta_{ij} |\hat{x}_i^j - \pi(R_i, T_i, X_j)|^2 \tag{37}$$

with $\theta_{ij} = 1$ **if point** j **is visible in image** i, $\theta_{ij} = 0$ else. Solution of this non-convex problem can be computed by various iterative algorithms, most importantly a damped version of the Gauss-Newton algorithm called **Levenberg-Marquardt algorithm**. Bundle adjustment is typically initialized by an algorithm such as the eight-point or five-point algorithm.