Visual SLAM Tutorial: Bundle Adjustment

Frank Dellaert

June 27, 2014

1 Minimizing Re-projection Error in Two Views

In a two-view setting, we are interested in finding the most likely camera poses T_1^w and T_2^w, both in the space $SE(3)$ of 3D poses, and $3D$ points $P_j^w \in \mathbb{R}^3$, with $j \in 1..n$, which we can do by minimizing the re-projection errors

$$f(T_c^w, P^w; p) = \|h(T_c^w, P^w) - p\|_\Sigma^2$$

associated with each measurement $p \in \mathbb{R}^2$. Above, the **measurement function** $h : SE(3) \times \mathbb{R}^3 \to \mathbb{R}^2$ predicts the measurement p from the unknowns $T_c^w = (R_c^w, t_c^w)$ and P^w, and the notation $\|e\|_\Sigma^2 \stackrel{\Delta}{=} e^T \Sigma^{-1} e$ is shorthand for the squared Mahalanobis distance with covariance Σ.

The objective function is then the log-likelihood of the unknowns given all such measurements p in the two views. Under the Gaussian noise assumptions, this is exactly

$$E\left(T_1^w, T_2^w, \{P_j\}\right) \stackrel{\Delta}{=} \sum_{i=1,2} \sum_j \|h(T_i^w, P_j^w) - p_{ij}\|_\Sigma^2 \tag{1}$$

2 In a Linear World...

In a linear world, the measurement function h would be a linear function of the unknowns. For reasons that will become clear, let us introduce a 6DOF unknown ξ_i associated with camera i, and a 3DOF unknwon δ_j associated with point j. Then linearity implies

$$\hat{p}_{ij} = h_{ij}\left(\xi_i, \delta_j\right) \stackrel{\Delta}{=} F_{ij}\xi_i + G_{ij}\delta_j$$

and the objective function

$$E\left(\xi_1, \xi_2, \{\delta_j\}\right) \stackrel{\Delta}{=} \sum_{i=1,2} \sum_j \|F_{ij}\xi_i + G_{ij}\delta_j - b_{ij}\|_\Sigma^2 \tag{2}$$

where we also introduced b_{ij}, the "linear" measurement. Then we can easily re-write the objective function (2) as

$$E\left(\xi_1, \xi_2, \{\delta_j\}\right) \stackrel{\Delta}{=} \|Ax - b\|_{\Sigma'}^2 \tag{3}$$

with x and b vectorized forms of our unknowns and measurements, respectively, and Σ' a suitable defined (block-diagonal) covariance matrix.

3 Sparse Matters

The matrix A will be a block-sparse matrix [Hartley and Zisserman, 2004]. For example, with 3 points, we have

$$A = \begin{bmatrix} F_{11} & & G_{11} & & \\ F_{12} & & & G_{12} & \\ F_{13} & & & & G_{13} \\ & F_{21} & G_{21} & & \\ & F_{22} & & G_{22} & \\ & F_{23} & & & G_{23} \end{bmatrix}, \; x = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix}, \; b = \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \\ b_{14} \\ b_{15} \\ b_{16} \end{bmatrix}$$

A linear least-squares criterion of the form (3) is easy to solve: we just take the derivative wrt x and set to zero, obtaining the condition

$$\left(A'A\right)x = A'b, \tag{4}$$

also known as the **normal equations**. In the above, the symmetric matrix $A'A$ is the known as the **Hessian** of the system. Because A is sparse, it is imperative that, for the sake of computational efficiency, a sparse solver is used to solve for x, e.g., using GTSAM or cholmod. The latter is built into MATLAB and is automatically invoked simply by typing $x = A\backslash b$.

4 Back to Reality

In reality, the measurement function $h(T_c^w, P^w)$ is quite non-linear, and generates the predicted measurement $\hat{p}$ by first transforming the point P^w into camera coordinates P^c, as specified by the camera T_c^w, then projecting the point so obtained into the image:

$$\hat{p} = h\left(T_c^w, P^w\right) \stackrel{\Delta}{=} \pi\left(g\left(T_c^w, P^w\right)\right)$$

In detail, the point P^c in camera coordinates is obtained by the coordinate transformation g, as in

$$P^c = g\left(T_c^w, P^w\right) \stackrel{\Delta}{=} \left(R_c^w\right)^T \left(P^w - t_c^w\right) \tag{5}$$

Then the predicted image coordinates $\hat{p}$ are obtained by the camera projection function π

$$\hat{p} = (\hat{x}, \hat{y}) = \pi\left(X^c, Y^c, Z^c\right) \stackrel{\Delta}{=} \left(\frac{X^c}{Z^c}, \frac{Y^c}{Z^c}\right) \tag{6}$$

Above we assumed that the measurements p are pre-calibrated to **normalized image coordinates** $p = (x, y)$, as we are interested in the setting where the calibration matrix K and radial distortion function $f(r)$ are known.

So how can we get back to the easy, linear setting? Maybe we can do a Taylor expansion, as in

$$h(T_c^w + \xi, P^w + \delta) \approx h(T_c^w, P^w) + F\xi + G\delta.$$

However, here we run into trouble, as "+" is not defined for elements of $SE(3)$. The next section introduces a more general view of Taylor expansion that will work, however.

5 A Twist of Li(m)e

The notion of an incremental update to an element on a manifold $a \in \mathscr{M}$ is provided by the notion of a **tangent space** at a a, denoted $T_a\mathscr{M}$, which is a vector space $\mathbb{R}^n$ with n equal to the dimensionality of the manifold. Any vector $x \in T_a\mathscr{M}$ in the tangent space can be mapped back to the manifold, with $x = 0$ mapping to a. Hence, vectors x correspond to updates around a, and this provides a way for our linearized problem to update the solution on the manifold [Absil et al., 2007].

In the special case that the manifold is also a group, i.e., a **Lie group**, the mapping is provided by the exponential map. In what follows, I use the notation from [Murray et al., 1994], directly specialized to $SE(3)$: a vector $\xi \in \mathbb{R}^6$ is mapped to an element of the **Lie Algebra** $\mathfrak{se}(3)$, in this case a 4×4 matrix, which is then mapped to a Lie group element by matrix exponentiation:

$$T_c^w \leftarrow T_c^w \exp \hat{\xi}$$

where the "hat operator"$\hat{}$ is the mapping from $\mathbb{R}^6$ to $\mathfrak{se}(3)$:

$$\xi \triangleq \begin{bmatrix} \omega \\ v \end{bmatrix} \theta \rightarrow \hat{\xi} \triangleq \begin{bmatrix} [\omega]_\times & v \\ 0 & 0 \end{bmatrix} \theta$$

For $SE(3)$ the 6-dimensional quantity (ω, v) is also called a **twist**, whereas $\theta \in \mathbb{R}^+$ is the amount of twist applied, resulting in a finite **screw** motion. In general, the elements of the vector space isomorphic to the Lie algebra are called **canonical coordinates**. More details about the Lie algebra $\mathfrak{se}(3)$ and its corresponding exponential map can be found in Appendix A.

6 A Generalized Taylor Expansion

Equipped with this notion, we can generalize the Taylor expansion to locally approximate a function $f : G \rightarrow \mathbb{R}^m$, with G an n-dimensional Lie group, as follows:

$$f(ae^{\hat{\xi}}) \approx f(a) + f'(a)\xi$$

where $\hat{\xi} \in \mathfrak{g}$, the Lie algebra of G, and $f'(a)$ is an $m \times n$ **Jacobian matrix**, linearly mapping the n-dimensional exponential coordinates ξ to m-dimensional corrections on $f(a)$. Note that $f'(a)$ is itself a function of a, i.e., the linear mapping is in general different for different values of a.

In the image projection case, we need to similarly generalize the notion of **partial derivatives**, to approximate the two-argument measurement function h:

$$h(T_c^w e^{\hat{\xi}}, P^w + \delta) \approx h(T_c^w, P^w) + h^{[1]}(T_c^w, P^w)\xi + h^{[2]}(T_c^w, P^w)\delta$$

Above $h^{[1]}(.)$ and $h^{[2]}(.)$ are the 2×6 and 2×3 Jacobian matrices corresponding to the partial derivatives of h in its first and second arguments, respectively. Note that the incremental pose $\exp(\hat{\xi})$ is applied on the camera side, and hence the exponential coordinates $\xi = (\omega, v)$ are *expressed in the camera frame*.

7 Nonlinear Optimization

By defining $F_{ij} \triangleq h^{[1]}(T_i^w, P_j^w)$, $G_{ij} \triangleq h^{[2]}(T_i^w, P_j^w)$, and the linearization **prediction error** $b_{ij} = h(T_i^w, P_j^w) - p_{ij}$, we recover exactly the linear objective function in (2), which we can minimize for the optimal exponential coordinates ξ_i and δ_j.

Gauss-Newton nonlinear optimization simply iterates between linearization, solve, and update, until a predefined convergence criterion is stasfied:

1. Start with a good initial estimate $\theta^0 = T_1^w, T_2^w, \{P_j\}$
2. Linearize (1) around θ^{it} to obtain A and b
3. Solve for $x = \xi_1, \xi_2, \{\delta_j\}$ using the normal equations (4)
$$\left(A'A\right) x = A'b$$
where $A'A$ the Gauss-Newton approximation to the Hessian of the actual non-linear system.
4. Update the nonlinear estimate θ^{it+1} using the exponential map
   (a) $T_1^w \leftarrow T_1^w \exp \hat{\xi}_1$
   (b) $T_2^w \leftarrow T_2^w \exp \hat{\xi}_2$
   (c) $P_j \leftarrow P_j + \delta_j$
5. If not converged, go to 2.

8 Too much Freedom!

A problem that will pop up when using the Gauss-Newton for the two-view case is that the Hessian $A'A$ will be singular. The reason is that there is a so-called **gauge freedom**: the solution is not uniquely determined, as we can displace the cameras using an arbitrary, 7DOF similarity transform without affecting the objective function. There are several ways to get around this:

- Switch to a minimal representation of a scale-free relative transform between the two cameras. This works well for two cameras but is not so easy to apply in the multi-camera case.
- Use the so called "inner constraints" from photogrammetry, which constrain the centroid and scale of the solution. This generalizes to multiple cameras, but a downside is that the inner constraints destroy some of the sparsity of the system, which affects computational efficiency.
- Add prior terms on the first pose and the distance of the second pose from the first pose (e.g., unity). This works well and scales to multiple cameras, but in a monocular situation there will still be scale drift.
- Fuse in information from other sensors, e.g., IMU and/or GPS.

In the case of a calibrated stereo-rig the problem of scale disappears, but one still has to put a prior on or clamp down the first pose in order to eliminate the gauge freedom. A common but unadvisable way is to use the technique from the next section to make the problem "disappear".

9 A Matter of Trust

When far from the solution, the Hessian approximation $A'A$ might be valid for a very small region only, and taking "quadratic" steps as given by the normal equations will lead to divergence. A well known way to deal with this issue is to limit the steps based on the amount of "trust" we have in the current linearization, switching to a more cautious gradient descent approach when far from the local minimum.

The most well-known of these so-called "region trust" methods is **Levenberg-Marquardt** (LM), which solves for

$$\left(A'A + \lambda D\right) x = A'b,$$

where D is either the diagonal of A or the identity matrix. For the latter case, with λ high, the solution is given by

$$x = \lambda^{-1} A'b,$$

which is gradient descent with step size λ^{-1}. When λ is zero or small, LM reverts to quadratic Gauss-Newton steps. Simple implementations of LM start with high values of λ and gradually decrease it, backing off if this leads to steps with non-decreasing error. More sophisticated variants put a lot more work into finding a good λ at every step.

What LM is *not* designed for is masking out gauge freedoms, although in many implementations it is used that way: taking smaller steps when not needed will lead to slower convergence.

10 Finally, the Jacobians...

To apply all of the above, one last task has to be undertaken, and that is deriving the Jacobians of the measurement function h. Because h itself is the composition of the coordinate transformation g followed by the projection π, the following chain rules apply:

$$h^{[1]}(T_c^w, P^w) = \pi'(p_c) \cdot g^{[1]}(T_c^w, P^w)$$
$$h^{[2]}(T_c^w, P^w) = \pi'(p_c) \cdot g^{[2]}(T_c^w, P^w)$$

We now need to compute three Jacobian matrices:

1. The 2×3 Jacobian matrix $\pi'(X^c, Y^c, Z^c)$ of the projection function π is easily computed as

$$\pi'(p_c) = \frac{1}{Z^c} \begin{bmatrix} 1 & 0 & -X^c/Z^c \\ 0 & 1 & -Y^c/Z^c \end{bmatrix} = \frac{1}{z^c} \begin{bmatrix} 1 & 0 & -\hat{x} \\ 0 & 1 & -\hat{y} \end{bmatrix} \tag{7}$$

2. The 3×3 Jacobian matrix $g^{[2]}(T_c^w, P^w)$ of the coordinate transformation g with respect to a change δ in the point P^w is also easy to compute, as

$$g(T_c^w, P^w + \delta) = (R_c^w)^T (P^w + t_c^w) + (R_c^w)^T \delta$$

   is clearly linear in δ, and hence

$$g^{[2]}(T_c^w, P^w) = (R_c^w)^T \tag{8}$$

3. The change in pose is a bit more challenging. It can be shown, however, that

$$g\left(T_c^w e^{\hat{\xi}}, P^w\right) = P^c + P^c \times \omega - v$$

This is linear in both ω and v, and hence the 3×6 Jacobian matrix $g^{[1]}(T_c^w, P^w)$ of g with respect to a change in pose easily follows as

$$g^{[1]}(T_c^w, P^w) = \begin{bmatrix} [P^c]_\times & -I_3 \end{bmatrix} \tag{9}$$

In other words, rotating the camera frame is magnified by the "arm" of the point in camera coordinates, whereas moving the camera simply moves the point's coordinates in the opposite direction.

In summary, using Eqns. 7, 8, and 9, the derivatives of image projection are

$$h^{[1]}(T_c^w, P^w) = \frac{1}{Z^c} \begin{bmatrix} 1 & 0 & -\hat{x} \\ 0 & 1 & -\hat{y} \end{bmatrix} \begin{bmatrix} [P^c]_\times & -I_3 \end{bmatrix} \tag{10}$$

$$h^{[2]}(T_c^w, P^w) = \frac{1}{Z^c} \begin{bmatrix} 1 & 0 & -\hat{x} \\ 0 & 1 & -\hat{y} \end{bmatrix} (R_c^w)^T \tag{11}$$

Note the "arm" of P^c for rays near the optical axis is now nearly canceled by the inverse depth factor $1/z^c$ in $\pi'(P^c)$, i.e., we expect flow nearly proportional to the incremental rotation ω.

11 More Than One Camera

Many readers will have already figured out that the entire story above generalizes to many cameras almost at once. The general case is known as Structure from Motion (SfM) or bundle adjustment. The total re-projection error now sums over all measurements p_k

$$E\left(\{T_i^w\}, \{P_j\}\right) \stackrel{\Delta}{=} \sum_k \|h(T_{i_k}^w, P_{j_k}^w) - p_k\|_\Sigma^2 \tag{12}$$

where i_k and j_k encode the data association, assumed known here. Techniques such as Levenberg-Marquardt and obtaining a good initial estimate will be of greater importance for the many-cameras case, as the objective function above is very non-linear, especially when points move closer to image planes and "coross-over" to the wrong side, in which case local minimizers such as Gauss Newton and LM will almost always converge to a local minimum.

Another often used technique in batch SfM is the "Schur complement" trick, where in the linear step one first solves for the point updates δ_j as a function of pose updates $\{\xi_j\}$. However, the utility of that in an online, visual SLAM scenario is debatable - a discussion for a different part of the tutorial.

A 3D Rigid Transformations

The Lie group $SE(3)$ is a subgroup of the general linear group $GL(4)$ of 4×4 invertible matrices of the form

$$T \triangleq \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

where $R \in SO(3)$ is a rotation matrix and $t \in \mathbb{R}^3$ is a translation vector. An alternative way of writing down elements of $SE(3)$ is as the ordered pair (R,t), with composition defined as

$$(R_1,t_1)(R_2,t_2) = (R_1R_2, R_1t_2 + t_1)$$

Its Lie algebra $\mathfrak{se}(3)$ is the vector space of 4×4 twists $\hat{\xi}$ parameterized by the *twist coordinates* $\xi \in \mathbb{R}^6$, with the mapping [Murray et al., 1994]

$$\xi \triangleq \begin{bmatrix} \omega \\ v \end{bmatrix} \theta \rightarrow \hat{\xi} \triangleq \begin{bmatrix} [\omega]_\times & v \\ 0 & 0 \end{bmatrix} \theta = \sum_{i=1}^{6} \xi_i G^i$$

where the 4×4 matrices G are called the generators for $\mathfrak{se}(3)$. Note I follow a different convention from Murray and reserve the first three components for rotation, and the last three for translation. Hence, with this parameterization, the generators for $SE(3)$ are

$$G^1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} G^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} G^3 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$G^4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} G^5 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} G^6 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Applying the exponential map to a twist ξ, scaled with $\theta \in \mathbb{R}^+$, yields a screw motion in $SE(3)$:

$$T(t) = \exp\left(\hat{\xi}\theta\right)$$

A closed form solution for the exponential map is given in [Murray et al., 1994, page 42].

$$\exp\left(\widehat{\begin{bmatrix} \omega \\ v \end{bmatrix}} t\right) = \begin{bmatrix} e^{[\omega]_\times \theta} & (I - e^{[\omega]_\times \theta})(\omega \times v) + \omega\omega^T v\theta \\ 0 & 1 \end{bmatrix}$$

References

[Absil et al., 2007] Absil, P.-A., Mahony, R., and Sepulchre, R. (2007). *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, USA.

[Hartley and Zisserman, 2004] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition.

[Murray et al., 1994] Murray, R., Li, Z., and Sastry, S. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press.