Operations Research - Assignment

Sun Qinxuan

May 20, 2017

1 Introduction

The scan matching technique plays an important role in the mobile robot simultaneous localization and mapping (SLAM). The scan matching generally recovers the robot pose by finding a transformation that best aligns the two successive scans. The alignment of scans is usually achieved by minimizing the geometric error or photometric error between two scans. In this paper, we aim to use a sparse Levenberg Marquardt algorithm [1] to solve the scan matching problem. In the following, a formulation of the scan matching problem is given in Section 2, and a brief introduction of the sparse LM algorithm is presented in Section 3. The concrete implementation and some experimental results are shown in Section 4.

2 Scan Matching

2.1 Minimization with Known Correspondences

We are given two 3D point sets $\{{}^{r}p_i, i = 1, ..., N\}$ and $\{{}^{c}p_i, i = 1, ..., N\}$ with ${}^{r}p_i$ and ${}^{c}p_i$ representing a pair of corresponding points that satisfies

$$ {}^{c}p_i = R_{cr}{}^{r}p_i + t_{cr} + \epsilon_i \tag{1} $$

where R_{cr} is a 3×3 rotation matrix, t_{cr} is a translation vector and ϵ_i is a noise vector. Then we want to find R_{cr} and t_{cr} to minimize

$$ E^2 = \sum_{i=1}^{N} \left[{}^{c}p_i - (R_{cr}{}^{r}p_i + t_{cr}) \right]^T \Omega_i \left[{}^{c}p_i - (R_{cr}{}^{r}p_i + t_{cr}) \right]. \tag{2} $$

Noted that if the information matrix Ω_i is omitted, (2) turns into

$$ E^2 = \sum_{i=1}^{N} || {}^{c}p_i - (R_{cr}{}^{r}p_i + t_{cr}) ||^2. \tag{3} $$

It has been proved in [2], [3] and [4] that minimization of (3) can be solved analytically. But we will use the cost function in the form of (2) for the concern of the information of the point distribution, and adopt the nonlinear optimization method to solve the minimization problem.

2.2 Minimization with Unknown Correspondences

However, for most scan alignment problem, there is no correspondence before two sets of scan data. In this case, Iterative Closest Point (ICP) algorithm [5] is utilized. The key concept of the standard ICP algorithm can be summarized in two steps: (1) compute correspondences between two scans. and (2) compute a transformation which minimizes distance between corresponding points. The second step is related to the problem stated in Section 2.1.

The distance metric used in the closest point search process is usually the Euclidean distance. Since the scan data used in the scan matching is usually quite large. The brute force nearest neighbor search is too time-consuming to achieve the real time performance. So the kd-tree structure is often used in the look up of closest points and hence the speed of the whole scan matching process can be maintained.

3 Nonlinear Optimization

In this section, two of the most common iterative parameter minimization methods, i.e., Newton iteration and Levenberg-Marquardt iteration, are briefly introduced.

3.1 Gauss-Newton

Suppose we are given a model $\mathbf{y} = f(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^M$ is the parameter vector that is to be estimated and $\mathbf{y} \in \mathbb{R}^N$ is the measurement vector. A measured value of $\mathbf{y}$ is provided, and we wish to find the vector $\hat{\mathbf{x}}$ that most nearly satisfies this functional relation. That is to say, we seek the vector $\hat{\mathbf{x}}$ that minimizes

$$||\epsilon||^2 = ||f(\hat{\mathbf{x}}) - \mathbf{y}||^2. \tag{4}$$

Note that $f(\cdot)$ is a nonlinear function, we need to start with an initial estimated value $\mathbf{x}_0$ and proceed to refine the estimate under the assumption that the function $f(\cdot)$ is locally linear. Let ϵ_0 be $\epsilon_0 = f(\mathbf{x}_0) - \mathbf{y}$. Assume that the function $f(\cdot)$ can be approximated at $\mathbf{x}_0$ by $f(\mathbf{x}_0 + \delta\mathbf{x}) = f(\mathbf{x}_0) + J \cdot \delta\mathbf{x}$, where J is the Jacobian matrix $J = \partial f/\partial \mathbf{x}$. We aim to find a point $\mathbf{x}_1 = \mathbf{x}_0 + \delta\mathbf{x}$ which minimizes $f(\mathbf{x}_1) - \mathbf{y} = f(\mathbf{x}_0) + J\delta\mathbf{x} - \mathbf{y} = \epsilon_0 + J\delta\mathbf{x}$. It turns out to be a linear minimization problem, and the vector $\delta\mathbf{x}$ is obtained by solving the normal equation

$$J^T J\delta\mathbf{x} = -J^T\epsilon_0. \tag{5}$$

Thus, the solution vector $\hat{\mathbf{x}}$ for next iteration is obtained by

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \delta\mathbf{x}_i \tag{6}$$

Now we define the squared norm of the error ϵ as a scalar valued function $g(\mathbf{x})$

$$g(\mathbf{x}) = \frac{1}{2}||\epsilon(\mathbf{x})||^2 = \frac{1}{2}\epsilon(\mathbf{x})^T\epsilon(\mathbf{x}). \tag{7}$$

The optimization problem is to minimize $g(\mathbf{x})$ over all values of $\mathbf{x}$. Firstly, two assumptions are made: $g(\mathbf{x})$ has a well-defined minimum value, and we know an initial point $\mathbf{x}_0$ reasonably close to the minimum. Then $g(\mathbf{x})$ is expanded about $\mathbf{x}_0$ in a Taylor series as

$$g(\mathbf{x}_0 + \delta\mathbf{x}) = g(\mathbf{x}_0) + g_{\mathbf{x}}\delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^T g_{\mathbf{xx}}\delta\mathbf{x} + ... \tag{8}$$

To minimize this quantity w.r.t. $\delta\mathbf{x}$, we differentiate w.r.t. $\delta\mathbf{x}$ and set the derivative to zero, yielding the equation

$$g_{\mathbf{xx}}\delta\mathbf{x} = -g_{\mathbf{x}}. \tag{9}$$

In this equation, $g_{\mathbf{xx}}$ is the matrix of second derivatives, i.e., the Hessian of g and the vector $g_{\mathbf{x}}$ is the gradient of g. From equation (7) we know that

$$g_{\mathbf{x}} = \epsilon_{\mathbf{x}}^T\epsilon \tag{10}$$

and

$$g_{\mathbf{xx}} = \epsilon_{\mathbf{x}}^T\epsilon_{\mathbf{x}} + \epsilon_{\mathbf{xx}}^T\epsilon. \tag{11}$$

Under the assumption that $f(\mathbf{x})$ is linear, the second term on the right vanishes, leaving

$$g_{\mathbf{xx}} = \epsilon_{\mathbf{x}}^T\epsilon_{\mathbf{x}} = J^TJ. \tag{12}$$

This assumption is reasonable when the second term of (11) is small enough to be negligible compared to the first term. Actually in practice, the term multiplying the second derivative in (11) is the random measurement error. Hence for a successful model, this error tend to cancel out when summed over all the measurements. So this procedure in which J^TJ is used as an approximation for the Hessian is known as the Gauss-Newton method.

3.2 Gradient Descent

The negative gradient vector $-g_{\mathbf{x}} = -\epsilon_{\mathbf{x}}^T\epsilon$ defines the direction of most rapid decrease of the cost function. The strategy of moving iteratively in the negative gradient direction is known as gradient descent. The length of the step is often computed by carrying out a line search for the function minimum in the negative gradient direction. It can be written as $\lambda\delta\mathbf{x} = -g_{\mathbf{x}}$, where λ controls the step length.

Gradient descent by itself is not a very good minimization strategy due to the zig-zagging phenomenon when it is close to the minimum point. Thus the Levenberg-Marquardt method is introduced which is essentially a Gauss-Newton method that transitions smoothly to gradient descent when the Gauss-Newton updates fail.

3.3 Levenberg-Marquardt

In the Levenberg-Marquardt (LM) iteration method, the normal equations $J^TJ\delta\mathbf{x} = -J^T\epsilon$ are replaced by the augmented normal equations $(J^TJ + \lambda I)\delta\mathbf{x} = -J^T\epsilon$. Here I is the identity matrix, and the value of λ varies from iteration to iteration.

Solving the augmented normal equations yields an increment $\delta\mathbf{x}$. If $\delta\mathbf{x}$ leads to a reduction in the error, then the increment is accepted and λ is divided by a factor before the next iteration. On the other hand if it leads to an increased error, then λ is multiplied by the same factor and the augmented normal equations are solved again. This process continues until a value of $\delta\mathbf{x}$ is found that gives rise to a decreased error.

Consider what happens for different values of λ. When λ is very small, the method is essentially the same as Gauss-Newton iteration. If λ is large then the normal equation matrix is approximated by λI, thus the normal equations become $\lambda\delta\mathbf{x} = -g_{\mathbf{x}}$. Therefore, the LM algorithm moves seamlessly between Gauss-Newton iteration and the gradient descent approach. The Gauss-Newton procedure will cause rapid convergence in the

neighborhood of the solution, and the gradient descent approach will guarantee a decrease in the cost function when the going is difficult. As the λ becomes larger, the length of the increment step $\delta\mathbf{x}$ decreases and eventually it will lead to a decrease of the cost function.

3.4 Sparse LM

In the central step of LM algorithm, the solution of the normal equations has complexity N^3 in the number of parameters, and this step is repeated many times. Hence, LM algorithm is not very suitable for minimizing cost functions w.r.t. large numbers of parameters. However, for many estimation problems in the computer vision field, the normal equation matrix has a certain sparse block structure that one may take advantage of to realize time savings.

For instance, the reconstruction problem in the computer vision in which one has image correspondences across two or more views and wishes to estimate the camera parameters of all the cameras and also the 3D positions of all the points. In this case, the set of parameters may be divided up into two sets: a set of parameters describing the cameras, and a set of parameters describing the points. More precisely, the parameter vector $\mathbf{x} \in \mathbb{R}^M$ can be partitioned into parameter vectors $\mathbf{a}$ and $\mathbf{b}$ so that $\mathbf{x} = (\mathbf{a}^T, \mathbf{b}^T)^T$. And the measurement vector $\mathbf{y} \in \mathbb{R}^N$ consists of all the image point coordinates. In addition, taking the covariance matrix for the measurement vector into account, one seeks the set of parameters that minimize the squared Mahalanobis distance $||\epsilon||^2_{\Sigma_\mathbf{x}} = \epsilon^T \Sigma_\mathbf{x}^{-1} \epsilon$. Corresponding to the division of parameters $\mathbf{x} = (\mathbf{a}^T, \mathbf{b}^T)^T$, the Jacobian matrix $J = [\partial\hat{\mathbf{y}}/\partial\mathbf{x}]$ has a block structure of the form $J = [A|B]$, where the submatrices are defined by

$$A = [\partial\hat{\mathbf{y}}/\partial\mathbf{a}] \tag{13}$$

and

$$B = [\partial\hat{\mathbf{y}}/\partial\mathbf{b}]. \tag{14}$$

The set of equations $J\delta\mathbf{x} = \epsilon$ solved as the central step in the LM algorithm has the form

$$J\delta\mathbf{x} = [A|B] \begin{pmatrix} \delta\mathbf{a} \\ \delta\mathbf{b} \end{pmatrix} \epsilon. \tag{15}$$

Then, the normal equations $J^T\Sigma_\mathbf{x}^{-1} J\delta\mathbf{x} = J^T\Sigma_\mathbf{x}^{-1}\epsilon$ to be solved at each step of the LM algorithm are of the form

$$\begin{bmatrix} A^T\Sigma_\mathbf{x}^{-1}A & A^T\Sigma_\mathbf{x}^{-1}B \\ B^T\Sigma_\mathbf{x}^{-1}A & B^T\Sigma_\mathbf{x}^{-1}B \end{bmatrix} \begin{pmatrix} \delta\mathbf{a} \\ \delta\mathbf{b} \end{pmatrix} = \begin{pmatrix} A^T\Sigma_\mathbf{x}^{-1}\epsilon \\ B^T\Sigma_\mathbf{x}^{-1}\epsilon \end{pmatrix}. \tag{16}$$

In the LM algorithm, the diagonal blocks of the matrix are augmented by multiplying their diagonal entries by a factor $1 + \lambda$ for the varying parameter λ, which alters the matrices $A^T\Sigma_\mathbf{x}^{-1}A$ and $B^T\Sigma_\mathbf{x}^{-1}B$. The resulting matrices are denoted by $(A^T\Sigma_\mathbf{x}^{-1}A)^*$ and $(B^T\Sigma_\mathbf{x}^{-1}B)^*$.

For simplicity, (16) can be written in the form

$$\begin{bmatrix} U^* & W \\ W^T & V^* \end{bmatrix} \begin{pmatrix} \delta\mathbf{a} \\ \delta\mathbf{b} \end{pmatrix} = \begin{pmatrix} \epsilon_A \\ \epsilon_B \end{pmatrix}. \tag{17}$$

To solve these equations, both sides are multiplied on the left by

$$\begin{bmatrix} I & -WV^{*-1} \\ 0 & I \end{bmatrix}$$

resulting in

$$\begin{bmatrix} U^* - WV^{*-1}W^T & 0 \\ W^T & V^* \end{bmatrix} \begin{pmatrix} \delta\mathbf{a} \\ \delta\mathbf{b} \end{pmatrix} = \begin{pmatrix} \epsilon_A - WV^{*-1}\epsilon_B \\ \epsilon_B \end{pmatrix}. \tag{18}$$

The top half of this set of equations is

$$(U^* - WV^{*-1}W^T)\delta\mathbf{a} = \epsilon_A - WV^{*-1}\epsilon_B. \tag{19}$$

And the value of $\delta\mathbf{b}$ can be found by

$$V^*\delta\mathbf{b} = \epsilon_B - W^T\delta\mathbf{a}. \tag{20}$$

4 Implementation

In this section, we utilize the sparse LM algorithm to estimate the transformation between two successive scans captured by an RGB-D camera. However, not all the scan data is exploited. We first detect the edge information in the scan [6], then use the edge points in the ICP process to obtain the camera pose estimation.

4.1 3D Rigid Body Transformation

A 3D rigid body transform $T \in \mathbb{SE}(3)$ denotes rotation and translation in 3D space which is defined by

$$T = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix} \quad with \quad \mathbf{R} \in \mathbb{SO}(3) \; and \; \mathbf{t} \in \mathbb{R}^3. \tag{21}$$

During optimization, a minimal representation for the camera pose is required, which is given by the corresponding element $\xi \in \mathfrak{se}(3)$ of the associated Lie-algebra. Elements are mapped to $\mathbb{SE}(3)$ by the exponential map $T = \exp(\xi)$, and it's inverse is denoted by $\xi = log(T)$.

4.2 Optimization on Lie-Manifolds

In each iteration of ICP, (22) is minimized w.r.t. ξ.

$$E^2 = \sum_{i=1}^{N} \left[{}^c p_i - T_{cr}({}^r p_i)\right]^T \Omega_i \left[{}^c p_i - T_{cr}({}^r p_i)\right]. \tag{22}$$

Denote $r_i^2(\xi)$ as

$$r_i^2(\xi) = \left[{}^c p_i - T_{cr}({}^r p_i)\right]^T \Omega_i \left[{}^c p_i - T_{cr}({}^r p_i)\right]. \tag{23}$$

The Jacobian matrix required in the LM optimization is then computed as

$$J = \frac{\partial E(\xi)}{\partial \xi} = \sum_{i=1}^{N} \frac{\partial r_i(\xi)}{\partial \xi} = -\sum_{i=1}^{N} \left[{}^c p_i - T_{cr}({}^r p_i)\right]^T \Omega_i \frac{\partial T_{cr}({}^r p_i)}{\partial \xi} \tag{24}$$

where

$$\frac{\partial T_{cr}({}^r p_i)}{\partial \xi} = [I, -T_{cr}({}^r p_i)\wedge] \tag{25}$$

with $\mathbf{q}\wedge$ represent the skew symmetric matrix of vector $\mathbf{q}$.

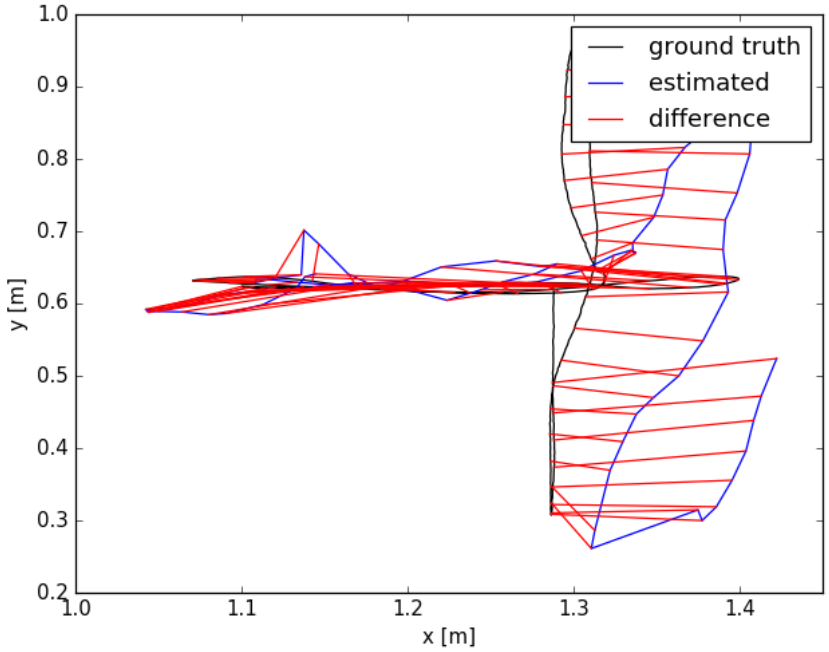

Figure 1: RMSE of ATE.

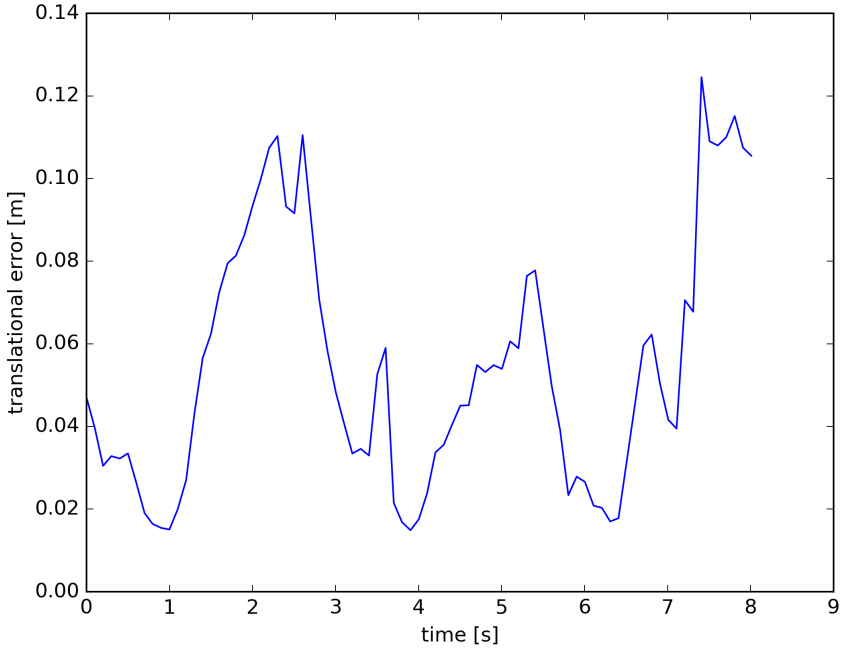

Figure 2: RMSE of RPE.

4.3 Experimental Results

All the experiments are performed on ubuntu 16.04 LTS, and the programs are written in C++. The g2o [7] framework is applied to achieve the sparse LM algorithm. The aforementioned frame-to-frame camera pose estimation is run as the RGB-D visual odometry on part of the Fr1-xyz image sequence of the Freiburg RGB-D benchmark [8]. The root mean square error (RMSE) of the absolute trajectory error (ATE) and the relative pose error (RPE) proposed in [8] are adopted as error metric to evaluate the pose estimation method. The ATE and RPE are shown in Figure 1 and Figure 2, respectively.

References

[1] M. I. A. Lourakis, Sparse non-linear least squares optimization for geometric vision, ECCV, 2010.

[2] K. S. Arun, T. S. Huang, And S. D. Blostein, Least-squares fitting of two 3-D point sets, IEEE Transactions On Pattern Analysis And Machine Intelligence, 1987.

[3] B. K. P. Horn, Closed-form solution of absolute orientation using orthonormal matrices, Journal of the Optical Society of America, 1988.

[4] B. K. P. Horn, Closed-form solution of absolute orientation using unit quaternions, Journal of the Optical Society of America, 1987.

[5] P. J. Besl, and N. D. McKay, A method for registration of 3-D shapes, IEEE Transactions On Patiern Analysis And Machine Intelligence, 1992.

[6] C. Choi, A. J. B. Trevor, and H. I. Christensen, RGB-D edge detection and edge-based registration, IROS, 2013.

[7] R. Kmmerle, G. Grisetti, H. Strasdat, K. Konolige and W. Burgard, G2o: A general framework for graph optimization, ICRA, 2011.

[8] J. Sturm, N. Engelhard, F. Endres, W. Burgard and D. Cremers, A benchmark for the evaluation of RGB-D slam systems, IROS, 2012.