Manifold Optimisation for Motion Factorisation

Appu Shaji
CSE , IIT Bombay
appu@cse.iitb.ac.in

Sharat Chandran
CSE , IIT Bombay
sharat@cse.iitb.ac.in

David Suter
ECSE , Monash University
d.suter@eng.monash.edu.au

Abstract

This paper presents a novel formulation for the popular factorisation based solution for Structure from Motion. Since our measurement matrices are populated with incomplete and inaccurate data, SVD based total least squares solution are less than appropriate. Instead, we approach the problem as a non-linear unconstrained minimisation problem on the product manifold of the Special Euclidean Group (SE_3). The restriction of the domain of optimisation to the SE_3 product manifold not only implies that each intermediate solution is a plausible object motion, but also ensures better intrinsic stability for the minimisation algorithm. We compare our method with existing state of art, and show that our algorithm exhibits superior performance.

1. Introduction

A favoured technique by researchers for computing Structure from Motion (SfM) solution is the factorisation scheme originally proposed in [8]. The central premise behind this approach is that a measurement matrix $\mathtt{P}$ populated by stacking up 2D point tracks of N feature points over a temporal window of F consecutive frames is rank limited. Further, an optimal r-rank approximation of $\mathtt{P}$ with respect to the Frobenius norm can be found out using a SVD-based scheme, by factoring $\mathtt{P}$ into to two sub-matrices, $\mathtt{R}$ (Rotation+Translation) and $\mathtt{S}$ (Shape) of size $2F \times r$ and $N \times r$ respectively.

$$\mathtt{P}_{2F\times N} = \mathtt{R}_{2F\times r}\mathtt{S}^T_{N\times r} \tag{1}$$

However, this method is conditioned on having a fully populated measurement matrix and the nature of noise present to be isotropic Gaussian. Unfortunately, in practice, most state-of-art trackers can only provide *incomplete* (occlusion) and *inaccurate* (transducer limits) point tracks if placed in an unstructured environment. Inspite of the more advanced schemes, and robustification techniques introduced subsequently, filling the incomplete data still remains a taxing problem to handle.

More promising schemes usually find the minimiser of *weighted reprojection error*, given by

$$f(\mathtt{R},\mathtt{S}) = ||\mathtt{W}\odot(\mathtt{P}-\mathtt{R}\mathtt{S}^T)||^2_{\text{Frobenius}} \tag{2}$$

Here $\odot$ is the Hadamard product ($\mathtt{C} = \mathtt{A}\odot\mathtt{B} : c_{ij} = a_{ij}b_{ij}$), and $\mathtt{W}$ is a *weight* matrix of the same size as $\mathtt{P}$. Zeros correspond to the missing elements of $\mathtt{P}$ and a non-zero entry is a normalised trust factor for the corresponding observation.

Unfortunately, it turns out that reprojection error based objective functions are optimal merely from a statistical point of view, but not from a geometrical point of view ([4], also see §Sec. 4). Besides, previous empirical studies [1] have indicated that in most cases the error surfaces swept by Eq. (2) have multiple local minima and are not well behaved. Hence, these methods are dependent on the presence of good initial conditions, and might converge to a local minima or even to an infeasible solution if we do not reinitialise.

Our Contribution: The central observation of this paper is that the underlying parameter space of the matrix $\mathtt{R}$ is constrained and is given by the *product manifold* of the Special Euclidean (SE_3) group. We can not only alleviate poor convergence properties of Eq. (2), but also enforce the proper geometric structure on Eq. (2), by the following minimisation reformulation:

$$\begin{aligned} \min_f \quad & f(\mathtt{R},\mathtt{S}) = ||\mathtt{W}\odot(\mathtt{P}-\mathtt{R}\mathtt{S}^T)||^2_{\mathtt{F}} \\ \text{subject to} \quad & \mathtt{R} \in \underbrace{SE_3\times\cdots\times SE_3}_{\text{F times}} \end{aligned} \tag{3}$$

Standard constrained optimisation approaches like Lagrange multipliers are not suitable for our task. Even though they enforce constraints by clever algebraic manipulations (using projection operators) they are unmindful to the geometric properties of the constraint set. For example, while using an quaternion parametrisation, a Lagrange multiplier has to added to ensure unit norm constraint. This constraint is purely algebraic and synthetic in nature. An axis-angle or infinitesimal rotation based parameterisation [9] merely gives us a first order linearlisation of the rotation matrix. The ambient structure of the manifold is not accounted for when calculating the derivatives of the cost function.

In this paper, we solve Eq. (3) by making use of the recently proposed geometric optimisation technique [3], which is able to perform the minimisation in Eq. (3) *directly on the constraint manifold*. Our novel algorithm is an iterative biquadratic algorithm, where $\mathtt{R}$ is updated using Gauss/Newton update on the smooth manifold formed by the N-fold product of special Euclidean group, whereas $\mathtt{S}$ is updated using a second order update [6].

Notation: All matrices are type-set with capital letters with type writer font (e.g., $\mathtt{A}$) , vectors with bold serif font (e.g., $\mathbf{a}$) and scalars with normal math fonts (e.g., a). If $\mathtt{A}$ is a $m \times n$ matrix, then $\text{vec}(\mathtt{A})$ is a $mn \times 1$ vector formed by writing down the columns of $\mathtt{A}$ one at a time. $\otimes$ indicates the Kronecker product.

978-1-4244-2175-6/08/$25.00 ©2008 IEEE

2 Our Method

The algorithm presented in this paper uses a coordinate descent optimisation scheme, where rotation ($\mathtt{R}$) and shape ($\mathtt{S}$) are updated alternatively while keeping the other term constant. Our point of departure with other coordinate approaches used to solve the SfM factorisation problem is that both the $\mathtt{R}$ and $\mathtt{S}$ updates are quadratic and not linear. This bi-quadratic updates do more justice to the cost functions defined in Eq. (2) and Eq. (3) which are quartic in nature. The bilinear update previously reported in the literature [7] linearises Eq. (2) and gives an approximate solution only.

A summary of the algorithm is given below

Algorithm 1 Overview of our algorithm

1: Initialise $\mathtt{R}$ and $\mathtt{S}$ using the method given in [8]
2: **repeat**
3: Minimise $f_{\mathtt{S}}(\mathcal{R}) = \mathrm{vec}^T(\mathcal{R})\mathtt{A}\,\mathrm{vec}(\mathcal{R}) + \mathtt{B}\,\mathrm{vec}(\mathcal{R}) + C$ subject to $\mathcal{R} \in SE_3^F$ and assuming $\mathtt{S}$ is constant by using the method explained in §Sec. 3, where

$$\mathtt{A} = (\Gamma \otimes \mathtt{S}^T)\mathtt{Q}(\Gamma^T \otimes \mathtt{S}) \quad \mathtt{B} = 2\,\mathrm{vec}(\mathtt{P}^T)\mathtt{Q}(\Gamma^T \otimes \mathtt{S})$$
$$\mathtt{C} = \mathrm{vec}^T(\mathtt{P}^T)\mathtt{Q}\,\mathrm{vec}(\mathtt{P}^T) \quad \mathtt{Q} = \mathrm{diag}(\mathrm{vec}(\mathtt{W}^T)\,\mathrm{vec}^T(\mathtt{W}^T)$$
$$\mathtt{R} = \Gamma^T\mathcal{R}^T \quad \Gamma^T = I_F \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

4: Update Shape parameters by minimising $f_{\mathtt{R}}(\mathtt{S}) = \left\|\mathrm{vec}(\mathtt{H}) \odot \left(\mathrm{vec}(\mathtt{P}) - (\mathtt{I}_n \otimes \mathtt{R})\,\mathrm{vec}(\mathtt{S}^T)\right)\right\|_F^2$ using Wiberg update [6]. {Due to space limitations we do not further develop shape updates here. More details are available at `http://www.cse.iitb.ac.in/appu/icprAM/`}
5: **until** convergence

3 Optimisation on a Manifold

We address the rotation update as an unconstrained optimisation problem on a constraint manifold (product manifold of special Euclidean manifold in our case). A traditional unconstrained or constrained optimisation methods perform searches in $\mathbb{R}^N$ where N is the dimensionality of search space, using iterative steps of the form $\mathbf{x}_{k+1} = \mathbf{x}_k + a\omega_K$, where $\mathbf{x}_k$ is the k-th iterate, a is a positive scalar, and ω_k is the descent direction. The descent directions are computed using first and (possibly) second order derivatives at the current location. To generalise the optimisation to manifolds. we need mechanism to calculate the gradient and Hessian on manifold. Further, while generalising the Newton update, we must ensure that any iterate $\mathbf{x}_{k+1}$ should lie on the manifold surface.

In our method, at each iterate, an Euclidean *local* parameterisation of the manifold is constructed (Eq. (5)). We carry out the optimisation of the local cost function in that parameter space. The descent direction and the step size are computed with the aid of the local versions of the Jacobian and Hessian (see Eq. (7)). We then project the optimal vector back to the manifold (see Fig. 1).

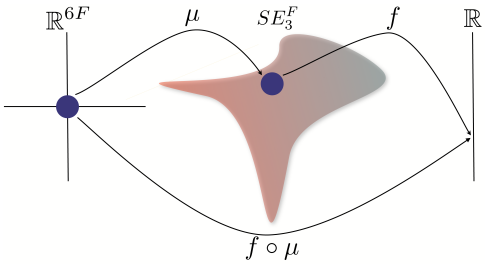


Figure 1. The mapping $\mu_{\mathcal{R}}$ is the local parametrisation of SE_3^F around point $\mathcal{R}$ such that $\mu_{\mathcal{R}}(0) = \mathcal{R}$. f is a smooth function defined on SE_3^F and $f \circ \mu_{\mathcal{R}}$ is f expressed in local parameter space $\mathbb{R}^{6F}$

The matrix $\mathtt{R}$ in Eq. (3) comprises of stacked motion matrices $\in SE_3$ (Special Euclidean space). Hence, the space of $\mathtt{R}$ matrices build from observations from F frames is a subspace on the SE_3^F product manifold. SE_3 forms a Lie group of six dimensions with the usual matrix multiplication. Manifolds induced by Lie Groups have the added advantage that they are smooth and their Lie algebra describes the tangent space at the identity of the matrix Lie group.

The Lie algebra of SE_3 is denoted by $\mathfrak{se}_3$. There is a well know isomorphism from $\mathbb{R}^6$ to $\mathfrak{se}_3$ given by the mapping Ω

$$\Omega(x) = \begin{bmatrix} [\omega]_\times & \mathbf{v} \\ 0 & 0 \end{bmatrix} \text{ where } \omega, \mathbf{v} \in \mathbb{R}^3, [\omega]_\times \text{is a skew symmetric matrix}$$

Tangent Space: It can be trivially show that the affine tangent space of a point $\mathcal{R} \in SE_3^F$ is given by

$$T_{\mathcal{R}}^{\mathrm{aff}} SE_3^F = \mathcal{R} + \mathcal{R}\tilde{\Omega}, \text{ where } \tilde{\Omega} = \Omega_1 \oplus \Omega_2 \oplus \cdots \oplus \Omega_N,$$
$$\Omega_i \in \mathfrak{so}_3 \text{ and } \oplus \text{ denotes the direct sum operator}$$

Local Parametrisation: Let $N(O) \subset \mathbb{R}^{6F}$ denote a sufficiently small open neighbourhood of the origin in $\mathbb{R}^{6F}$. Then the exponential mapping

$$\mu : N(0) \subset \mathbb{R}^{6F} \rightarrow SE_3^F, x \rightarrow \mathtt{R}e^{\Omega(x)} \tag{4}$$

is a local diffeomorphism from $N(0)$ onto a neighbourhood of $\mathtt{R}$ in SE_3.

The cost function $f_{\mathtt{S}}(\mathtt{R})$ (Alg. 1 , step 3) at $\mathcal{R} \in SE_3^F$ (see Fig. 1) expressed in local parameter space using the smooth local parametrisation $\mu_{\mathcal{R}}$ is given by

$$f \circ \mu_{\mathcal{R}}(\omega) = \mathrm{vec}^T(\mathcal{R}e^{\tilde{\Omega}(\omega)})\mathtt{A}\,\mathrm{vec}(\mathcal{R}e^{\tilde{\Omega}(\omega)}) - \mathtt{B}\,\mathrm{vec}(\mathcal{R}e^{\tilde{\Omega}(\omega)}) + \mathtt{C} \tag{5}$$

Notice that this function is no longer quadratic. We obtain a quadratic approximation of Eq. (5) by taking its second order Taylor approximation.

$$T(f \circ \mu_{\mathcal{R}}) : (f \circ \mu_{\mathcal{R}})(t\mu) + \frac{\mathrm{d}(f \circ \mu_{\mathcal{R}})(t\mu)}{\mathrm{d}t} + \frac{1}{2}\frac{\mathrm{d}^2(f \circ \mu_{\mathcal{R}})(t\mu)}{\mathrm{d}t^2}\Big|_{t=0} \tag{6}$$

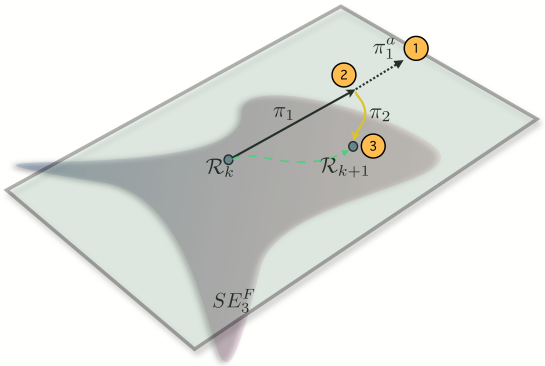


Figure 2. Our algorithm first maps a point $\mathbb{R}_K \in SE_3^K$ to an element of the affine tangent space $T^{\text{aff}}_{\mathcal{R}_k} SE_3^K$ via π_1 followed by step π_2 to project the vector back to manifold

Jacobian and Hessian for this quadratic approximation (after omitting the copious algebra) is given by

$$\nabla_{f\circ\mu_{\mathcal{R}}}(0) = \mathtt{J}^T(\mathtt{I}\otimes\mathtt{R}^T)(\mathtt{A}\,\mathrm{vec}(\mathtt{R}) - \frac{1}{2}\mathtt{B}^T) \quad (7)$$

$$\mathtt{H}_{f\circ\mu_{\mathcal{R}}}(0) = \mathtt{J}^T(\mathtt{I}\otimes\mathcal{R}^T)\mathtt{A}(\mathtt{I}\otimes\mathcal{R}^T)\mathtt{J} + \mathtt{J}^T(\mathtt{I}\otimes\mathcal{R}^T\mathtt{C})\mathtt{J} \quad (8)$$

$$\text{where } \mathrm{vec}(\tilde{\Omega}) = \mathtt{J}\mathbf{x}; \mathbf{x} = (\omega^T\mathbf{v}^T)^T$$

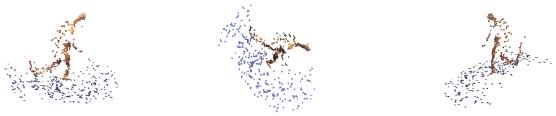
A pictorial representation of the algorithm is given in Fig. 2 The optimal descent direction (in spirit of the classical Newton's algorithm) is set to be $-\mathtt{H}^{-1}_{f\circ\mu_{\mathcal{R}}}\nabla(f\circ\mu_{\mathcal{R}})$, represented by π_1^a (step 1 in Fig. 2). This is followed by a one dimensional backtracking based inexact line search by maintaining the Wolfe condition [2] that ensures reduction in cost function (π_1; step 2 in Fig. 2). Once the descent direction and downhill step size is obtained we map the resulting point back to the manifold via the *push forward* to the manifold using the exponential map operator(π_2; step 3 in Fig. 2). The above algorithm has quadratic convergence near a critical point [3]. The complete mathematical exposition for the algorithm is given at `http://www.cse.iitb.ac.in/appu/icprAM/`

4 Experiments and Results

Simulations: All simulations are carried out in Matlab. Each trial consists of the following:

- Randomly assign nearest integer values in the range $(20, 100)$ to F, the number of frames and $(20, 100)$ to N, the number of points. Note: we enforce $N < F$.
- F different rotation matrices are created by randomly assigned Euler angles and translation components. These are stacked to form $\mathcal{R}_{4F\times 4}$ and $\mathtt{R}_{2F\times 4}$. Matrix $\mathtt{S}_{N\times 4}$ is populated with random data.

(a) Input Sequence: Frame 1, 9,18, 27 and 36



(b) Reconstructed shape rendered from novel viewpoints

Figure 3. Toy Dinosaur sequence reconstruction

- Set $\mathtt{P}$ as $\mathtt{R}\mathtt{S}^T$, Also record the ground truth $\mathtt{P}_{\text{ground truth}}$ as $\mathcal{R}\mathtt{S}^T$
- Mask the band diagonal entries ($\mathtt{W}$). The band width is randomly chosen between $(1, \min\left(\frac{2}{3}F, \frac{2}{3}N\right))$
- Perform the Optimisation Algorithm.
- We compute two error terms. The *cost function error* is given by the value of Eq. (2). The *ground truth error* measures the sum squared error between the recovered motion estimate and the actual $3D$-error, defined as $||\mathtt{P}_{\text{ground truth}} - \mathcal{R}_{\text{reconstructed}}\mathtt{S}^T_{\text{reconstructed}}||^2_F$. We are more interested in the ground truth error, rather than the cost function error, but the latter drives the convergence

Real Data: We test our method with the toy dinosaur data [1] mounted on a rigid turntable. The toy dinosaur is tracked with a standard KLT tracker. The observation matrix thus created is 72×319 with only 28% of the measurement matrix populated. Our reconstruction is shown in Fig. 3. The rendering method used is similar to the one proposed in [5]

Quantitative Analysis: The central purpose of our simulations are to verify our claim that the constrained optimisation specified in Eq. (3) minimised by an algorithm which is sensitive to the geometrical structure is a better formulation for solving factorisation based SfM rather than its unconstrained counterpart (Eq. (2)).

We compare our algorithm with the damped Newton algorithm introduced in [1] which is one of the best know min-

Matrix Size	SE3-I	DN-I	SE3-C	DN-C
25×25	0.2727	0.3590	0.2727	2.9695
50×50	1.4211	0.8226	2.8423	11.2063
100×100	5.0469	2.0734	10.0938	22.4077
125×125	12.0937	2.4959	24.1875	37.2671
150×150	76.3279	4.8076	76.3279	53.1265

Table 1. Timing Comparison (All entries are in seconds), SE3-I: Time per iteration for our method, DN-I: Time per iteration for the damped Newton method, SE3-C: Time till convergence for our method, DN-C: Time till convergence for the damped Newton method

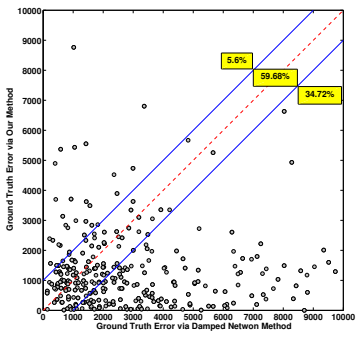
imiser of Eq. (2). We affine correct the output of the damped Newton algorithm so as to build proper rotation matrices using the method specified in [8]. 1250 different instances of the simulation are carried out. Both the methods are initialised by the same $\mathtt{R}$ and $\mathtt{S}^T$ matrices and a maximum iteration limit of 30 is specified.

The relative performance of the algorithms in terms of ground truth error are shown in Fig. 4(a). The scatter plot is divided into three regions, the region inside the two blue lines and the regions above and below it. The data points within the blue lines represent the instances when the performance of both the algorithm are similar (error within a standard deviation of 10%), while the region above it represents the instances when our algorithm perform worse than the damped Newton method. Notably the region below represents the instances when our method yields better solutions. It was observed that our method performed better in 34.72% instances, the performance of the algorithms were similar in 59.68% instances and 5.6% instances we got inferior results. The number of data points below the red dotted line (which gives a hard threshold) constituted 71.28% of the total instances.

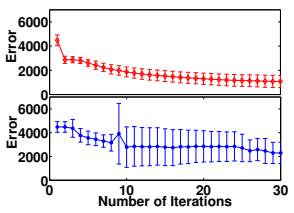
Fig. 4(b) which is an error-bar plot of Ground Truth Error for 50 different instances of the simulation. The top figure (data plotted in red) shows the behaviour of our algorithm whereas the bottom figure (data plotted in blue) shows the behaviour of the damped Newton algorithm. The vertical bars in the plot represents the standard deviation whereas the single continuous line represent the average error. We observe an asymptotic convergence pattern for our algorithm, with a relatively low band-limited set of standard deviations. However, the behaviour for the damped Newton case is erratic with high standard deviation terms.

Sensitivity Analysis: We fix $F \leftarrow 60$ and $N \leftarrow 40$. The percentage of missing components is varied from 5% to 30%. The simulation is repeated 100 times while fixing the value of missing components. An error bar plot with half of standard deviation as the vertical lines and average ground truth error value as the single continuous line is given in Fig. 4(c). It is observed that our method is relatively insensitive to missing data with respect to the ground truth error than the damped Newton data.

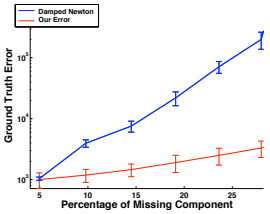
Timing and Convergence: Our algorithm usually converges within first 3-6 iterations, while the damped Newton case usually takes about 10-20. We state candidly that cost per iterate of our method is greater than the damped Newton case (Table 1). However this is compensated by the fact that the number of iterations required for convergence is much lower than the damped Newton case. For example, we have faster convergence timing in the first four cases at Table 1. The relatively poor running time in the last case is due to the exponential map operator, and also due to the larger matrix dimensions as the result of the Kronecker product.



(a) Relative performance of our method vs the damped Newton method.



(b) Behaviour of Ground Truth Error (Fifty instances). Bar indicates the standard deviation



(c) Sensitivity Analysis

Figure 4. Quantitative justification for our method.

5 Final Remarks

The best factorisation method for recovering three dimensions in the presence of incomplete and incorrect two-dimensional input data relies heavily on non-linear optimisation. In contrast with other methods that do not pay too much attention to the geometrical properties, we show how to perform the optimisation on the SE_3 manifold (induced by the rotation matrix). As a result, every step in the iterative process represents a valid geometrical solution.

References

[1] A. Buchanan and A. Fitzgibbon. Damped Newton algorithms for matrix factorization with missing data. *IEEE Conference on Computer Vision and Pattern Recognition*, 2:316–322, 2005.

[2] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2nd edition edition, 1987.

[3] P. Y. Lee. *Geometric Optimization for Computer Vision*. PhD thesis, Department of Information Engineering, Research School of Information Sciences and Engineering, Australian National University, 2005.

[4] Y. Ma, J. Košecká, and S. Sastry. Optimization criteria and geometric algorithms for structure from motion and structure estimation. *International Journal of Computer Vision*, 44(3):219–249, 2001.

[5] D. Martinec and T. Pajdla. 3d reconstruction by fitting low-rank matrices with missing data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 198–205, Washington, DC, USA, 2005. IEEE Computer Society.

[6] T. Okatani and K. Deguchi. On the wiberg algorithm for matrix factorization in the presence of missing components. *International Journal of Computer Vision*, 72(3):329–337, 2007.

[7] H. Shum, K. Ikeuchi, and R. Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(9):854–867, 1995.

[8] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, pages 137–154, 1992.

[9] B. Triggs, P. McLauchlan, R. I. Hartley, and F. A.W. Bundle adjustment - a modern synthesis. *Vision Algorithms: Theory and Practice, International Workshop on Vision Algorithms*, pages 298–373, 1999.