Lie-Algebraic Averaging For Globally Consistent Motion Estimation

Venu Madhav Govindu *
HIG-25, Simhapuri Layout
Visakhapatnam, AP 530047
INDIA
Email : venu@narmada.org

Abstract

While motion estimation has been extensively studied in the computer vision literature, the inherent information redundancy in an image sequence has not been well utilised. In particular as many as $\frac{N(N-1)}{2}$ pairwise relative motions can be estimated efficiently from a sequence of N images. This highly redundant set of observations can be efficiently averaged resulting in fast motion estimation algorithms that are globally consistent. In this paper we demonstrate this using the underlying Lie-group structure of motion representations. The Lie-algebras of the Special Orthogonal and Special Euclidean groups are used to define averages on the Lie-group which in turn gives statistically meaningful, efficient and accurate algorithms for fusing motion information. Using multiple constraints also controls the drift in the solution due to accumulating error. The performance of the method in estimating camera motion is demonstrated on image sequences.

1. Introduction

The development of a variety of geometric methods for estimating camera motion from video sequences [10] has been one of the most significant advances in computer vision. On the one hand, fast algorithms based on algebraic elimination of the structure variables have been developed to solve for the geometry of upto four views. However being restricted to few views, these methods have limited accuracy. On the other hand, the "optimal method" based on bundle-adjustment that minimises the distance between observed feature points and their estimated reprojections [9, 19] is computationally expensive as it involves non-linear optimisation of an objective function that solves for both motion and structure and requires a good initialisation.

The method described in this paper overcomes some of the above mentioned accuracy-complexity limitations by making efficient use of the existing information redundancy in a sequence to derive fast and accurate motion estimates. It is based on the observation that for a sequence of N images the global motion of the entire sequence is parametrised by $N-1$ independent motions. However in this sequence there exist *as many as* $\frac{N(N-1)}{2}$ image pairs each of which provides a constraint on the global motion. Moreover the two-view epipolar geometry (i.e. each constraint) can be estimated very efficiently. Consequently, we have an over-determined system of equations that can be "averaged" in a principled and efficient manner by using the Lie algebra of the representations. Some other methods for multi-frame motion estimation are [20], [18] where structure and motion are solved for simultaneously using a rank-based factorisation technique which usually require points to be tracked through the entire sequence. However our method operates on the available information and does not suffer from this drawback. Also by using the epipolar geometry between image pairs we eliminate the dependence on the unknown structure points. By averaging the available redundancy of motion information one can enforce global consistency on the solution which is particularly useful for long or closed sequences. Without the global consistency that our method automatically enforces, the inter-frame errors would increase as the length of the sequence increases. As will be noticed in the results presented this gives rise to accurate solutions.

In Sec. 2, we briefly present elementary ideas in Lie group theory and in Sec. 2.1 an algorithm for averaging elements of a Lie group is described. In Sec. 3 we describe our scheme for averaging relative motions to solve for global motion and present the main contribution of this paper, a Lie-algebraic scheme for efficiently averaging relative motions. Sec. 4 describes the results of applying this methods to motion estimation for real image sequences.

2. Lie Groups

As the requisite background for this paper, in this section we provide a very elementary, informal introduction to Lie

*Thanks to Neeta Deshpande for logistical support and Cafe Ananta, Taleigao for hot samosas during the writing of this paper.

groups. The structure of Lie groups forms the basis of our averaging method. For an excellent exposition on the use of Group Theory in Computer Vision, see [12]. A group G is a set whose elements satisfy the relationships

$$\begin{aligned} X.(Y.Z) = (X.Y).Z(associativity) \\ \exists E \ni X.E = E.X = X(identity) \\ \exists X^{-1} \ni X.X^{-1} = X^{-1}.X = E(inverse) \end{aligned}$$

A Lie group is a group for which the operations $X \times Y \mapsto XY$ and $X \mapsto X^{-1}$ are differentiable mappings. Intuitively, Lie groups can be locally viewed as topologically equivalent to the vector space, $\mathbb{R}^n$. Thus the *local* neighbourhood of any group element G can be adequately described by its tangent-space. The elements of this vector space form a Lie algebra $\mathfrak{g}$[1]. The Lie algebra $\mathfrak{g}$ is a vector space equipped with a bilinear operation $[.,.] : \mathfrak{g} \times \mathfrak{g} \rightarrow \mathfrak{g}$ known as the Lie bracket. The bracket satisfies the relationships

$$\begin{aligned} [\mathbf{x}, \mathbf{y}] = -[\mathbf{y}, \mathbf{x}] \, (anti - symmetry) \\ [\mathbf{x}, [\mathbf{y}, \mathbf{z}]] + [\mathbf{y}, [\mathbf{z}, \mathbf{x}]] + [\mathbf{z}, [\mathbf{x}, \mathbf{y}]] = 0 \, (Jacobi \ identity) \end{aligned}$$

All finite-dimensional Lie groups have matrix representations and the bracket in this case is the commutator operation $[\mathbf{x}, \mathbf{y}] = \mathbf{xy} - \mathbf{yx}$. The Lie algebra and the associated Lie group are related by the exponential mapping. Thus for matrix representations we have $exp(\mathbf{x}) = \sum_{k=0}^{\infty} \frac{\mathbf{x}^k}{k!}$. Hence, for a three-dimensional rotation by angle $\boldsymbol{\omega}$, we have $\mathbf{R}(\boldsymbol{\omega}) = e^{[\boldsymbol{\omega}]_\times}$ where $[\boldsymbol{\omega}]_\times$ is the skew-symmetric matrix representation of $\boldsymbol{\omega}$. Here $\mathbf{R} \in \mathbf{SO(3)}$ the Special Orthogonal group and $[\boldsymbol{\omega}]_\times \in \mathfrak{so}(3)$ the associated Lie algebra. Also the inverse mapping from the Lie algebra to the Lie group exists and is the logarithm function, i.e. $[\boldsymbol{\omega}]_\times = \log(\mathbf{R})$. It is appropriate to mention here the other Lie group we will use in this paper, the Special Euclidean Group $\mathbf{SE(3)}$ which represents the Euclidean transformation of rotation followed by translation. We shall represent elements of this group by $\mathbf{M}$ and elements of the associated Lie algebras $\mathfrak{m} \in \mathfrak{se}(3)$.

For non-commutative Lie groups, the usual exponential relation $e^{\mathbf{x}} e^{\mathbf{y}} = e^{\mathbf{x}+\mathbf{y}}$ does not hold. The equivalent mapping is defined by $BCH : \mathfrak{g} \times \mathfrak{g} \mapsto \mathfrak{g}$, i.e. $e^{\mathbf{x}} e^{\mathbf{y}} = e^{BCH(\mathbf{x},\mathbf{y})}$. The mapping $BCH(.,.)$ is defined by the Baker-Campbell-Hausdorff (BCH) formula [21] as a series

$$BCH(\mathbf{x}, \mathbf{y}) = \mathbf{x} + \mathbf{y} + \frac{1}{2}[\mathbf{x}, \mathbf{y}] + \frac{1}{12}[\mathbf{x} - \mathbf{y}, [\mathbf{x}, \mathbf{y}]] + \mathcal{O}(|(\mathbf{x}, \mathbf{y})|^4)$$

[1]Throughout we shall use capital letters to represent the groups and small letters to represent Lie algebras

With a view to the use of Lie algebra for defining an average later we note that the BCH formula is the intrinsic (Riemannian) distance on the manifold representing the group. For example, for rotations $\boldsymbol{\omega_1}$ and $\boldsymbol{\omega_2}$, $BCH(\boldsymbol{\omega_2}, -\boldsymbol{\omega_1})$ represents the rotation ("distance") that will take us from $\boldsymbol{\omega_1}$ to $\boldsymbol{\omega_2}$.

2.1. Averaging on the Lie Group

In this section we will describe the notion of an average of elements of a Lie group. It may be recalled that for a vector space, the sample mean or average of a set $\{\mathbf{X}_1, \cdots, \mathbf{X}_N\}$ is given by $\bar{\mathbf{X}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{X}_i$ which is the variational minimiser of the "deviation" $\sum_{i=1}^{N} d^2(\mathbf{X}_i, \bar{\mathbf{X}}) = \sum_{i=1}^{N} (\mathbf{X}_i - \bar{\mathbf{X}})^2$. However such a notion cannot be applied directly to elements of a group since a group manifold is not equivalent to a vector space. For example, the arithmetic average of two rotation matrices is not necessarily a valid rotation matrix.

If we view elements of a group G as being embedded in a real, vector space the sample average is denoted as the *extrinsic* average. Here, the group is first embedded in a Euclidean space $\phi : G \mapsto \mathbb{R}^n$ which induces a metric on the space and the sample average can be defined as $\overline{\phi(\mathbf{X})} = \frac{1}{N} \sum_{k=1}^{N} \phi(\mathbf{X}_k)$. However this sample average is not necessarily an element of the group G and we need to project it onto the manifold G in an optimal sense, $\bar{\mathbf{X}} = \mathcal{P}(\overline{\phi(\mathbf{X})})$. An instance of such a projection in computer vision is the eight-point algorithm [8]. Another instance of extrinsic averaging is the SVD-based scheme for averaging rotations presented in [2].

On the other hand if we define $d(.,.)$ as the *intrinsic* distance between points on a manifold (i.e. the Riemannian distance) then the "true" intrinsic average can be defined as

$$\mu = \arg \min_{\mathbf{X} \in G} \sum_{k=1}^{N} d^2(\mathbf{X}_k, \mathbf{X})$$

Here, the intrinsic average is computed by measuring the Riemannian distance between elements of the group and is automatically an element of the group G. In general, the intrinsic average is preferable over the extrinsic average but is often hard to compute due to the non-linearity of the Riemannian distance function $d(.,.)$ and the need to parametrise the group manifold G. However as we will see here, for matrix Lie groups the intrinsic average can be computed efficiently. For pedagogical purposes, the method outlined in the remainder of this subsection has been adopted from [5].

It will be recalled that for matrix groups, the Riemannian distance is defined by the matrix logarithm operation, i.e.

for matrix group elements $\mathbf{X}$ and $\mathbf{Y}$ we have [2]

$$d(\mathbf{X}, \mathbf{Y}) = ||\log(\mathbf{Y}\mathbf{X}^{-1})||$$

Using the BCH formula this distance can be approximated as

$$\begin{aligned} d(\mathbf{X}, \mathbf{Y}) &= ||\log(\mathbf{Y}\mathbf{X}^{-1})|| \\ &\approx ||\log(\mathbf{Y}) - \log(\mathbf{X})|| = ||\mathbf{y} - \mathbf{x}|| \end{aligned} \tag{4}$$

where $\mathbf{x}$ and $\mathbf{y}$ are logarithms of matrices $\mathbf{X}$ and $\mathbf{Y}$ respectively. Thus we note that the Riemannian distance between elements of a Lie group is now approximated by the "Euclidean" distance in its Lie algebra, i.e. its a tangent space approximation. For a set of group elements $\{\mathbf{X}_1, \cdots, \mathbf{X}_N\}$ the minimiser of $\sum_{i=1}^{N} ||\mathbf{X}_i - \bar{\mathbf{X}}||^2$ is estimated from the sample average of the Lie algebra $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$. Given this estimate of the average $\mu = exp(\frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i)$ we can remap the samples by left-multiplying by the inverse of μ. In other words we have $\Delta\mathbf{X}_i \leftarrow \mu^{-1}\mathbf{X}_i$ which loosely speaking is the residues of the original samples after we have "subtracted" μ. This operation can be repeated till the estimate converges to a local minima and can be summarised in the following algorithm.

A1 : Algorithm for Intrinsic Average

Input : $\{\mathbf{X}_1, \cdots, \mathbf{X}_N\} \in G$ (Matrix Group)
Output : $\mu \in G$ (Intrinsic Average)
Initialise : $\mu = \mathbf{I}$ (Identity)
Do
$\Delta\mathbf{X}_i = \mu^{-1}\mathbf{X}_i$
$\Delta\mathbf{x}_i = log(\Delta\mathbf{X}_i)$
$\Delta\mu = exp(\frac{1}{N}\sum_{i=1}^{N} \Delta\mathbf{x}_i)$
$\mu = \mu\Delta\mu$
$Repeat\ till\ ||\Delta\mu|| < \epsilon$

Following arguments in [1] it can be seen that this iteration is similar in spirit to a gradient-descent algorithm and always converges to a local minima. Also at every stage of this algorithm, we are always *on* the manifold and are never outside it which is often preferable compared to using an extrinsic average. Also the first-order approximation of the sample average is an aspect of the algorithm and does not affect the location of the convergence point. In passing we note that one could conceivably use the higher-order terms of the BCH formula in the estimation at each iteration, resulting in a more complex formulation that is a better approximation. However this does not affect the location of the fixed point.

[2] The norm used here is the Frobenius norm

2.2. Lie Algebras of $\mathbf{SO(3)}$ and $\mathbf{SE(3)}$

In this subsection we state the Lie algebras of the Rotation and Euclidean groups that we shall use for the motion estimation algorithms. Any three-dimensional rotation $\mathbf{R}$ is an element of the Special Orthogonal Group $\mathbf{SO(3)}$ and satisfies the constraint $\mathbf{R}\mathbf{R}^T = \mathbf{I}$. The exponential and logarithmic mappings are given by the familiar Rodrigues formula [14]. Incidentally, it is of particular interest to note that the BCH formula for elements of $\mathbf{SO(3)}$ has a closed form expression [17, 3]. For rotations $\boldsymbol{\omega_1}$ and $\boldsymbol{\omega_2}$, the BCH form is given by $BCH(\boldsymbol{\omega_1}, \boldsymbol{\omega_2}) = \alpha\boldsymbol{\omega_1} + \beta\boldsymbol{\omega_2} + \gamma\,\boldsymbol{\omega_1} \times \boldsymbol{\omega_2}$, where the scalars α, β and γ are functions of $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$. Consequently in the algorithm for intrinsic averaging of rotations, we need not compute the logarithms and exponentials at each iteration. Instead we can directly work in the Lie algebras till the algorithm terminates which results in faster computations.

In turn Euclidean motions, $\mathbf{M} \in SE(3)$ are represented by 4×4 matrices

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{5}$$

where $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$ are the three-dimensional rotation and translation respectively. The action of this transformation is to rotate a point followed by a translation. The logarithm of $\mathbf{M}$ is an element of the the Lie algebra $\mathfrak{m} \in \mathfrak{se}(3)$ and is given by

$$\mathfrak{m} = \begin{bmatrix} \boldsymbol{\Omega} & \mathbf{u} \\ 0 & 0 \end{bmatrix} \tag{6}$$

where skew-symmetric matrix $\boldsymbol{\Omega} \in \mathfrak{so}(3)$ and $\mathbf{u} \in \mathbb{R}^3$. Since the exponential map relates $\mathfrak{m}$ to the matrix $\mathbf{M}$ we have

$$\mathbf{M} = exp(\mathfrak{m}) = \sum_{k=0}^{\infty} \frac{\mathfrak{m}^k}{k!} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

By writing out the series expression one can show that the following relationships hold

$$\begin{aligned} \mathbf{R} &= exp(\boldsymbol{\Omega}) \\ \mathbf{t} &= \mathbf{P}\mathbf{u} \\ for\ \mathbf{P} &= \mathbf{I} + \frac{(1 - \cos\theta)}{\theta^2}\boldsymbol{\Omega} + \frac{(\theta - \sin\theta)}{\theta^3}\boldsymbol{\Omega}^2 \end{aligned} \tag{7}$$

where $\theta = \sqrt{\frac{1}{2}tr(\boldsymbol{\Omega}^T\boldsymbol{\Omega})}$.

3. Globally Consistent Motion Estimation

In this section we develop our solution for multi-frame motion estimation. The following analysis applies equally to

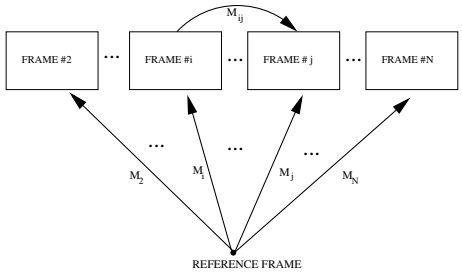


Figure 1: The relative motions are estimated from the data. The global motion with respect to the first frame is estimated by averaging the over-determined set of relative motion constraints.

both rotation and Euclidean motion estimation and a linear solution for this formulation was described earlier in [7]. For N images, the globally consistent motion can be described by $N-1$ motions. In general, we pick the first image as the reference frame and the rest of the sequence is estimated with respect to this reference frame. We denote the motion between frame i and the reference frame as $\mathbf{M}_i$, and the relative motion between two frames i and j as $\mathbf{M}_{ij}$, where $\mathbf{M}_{ij} = \mathbf{M}_j {\mathbf{M}_i}^{-1}$. This relationship captures the notion of "consistency", i.e. the composition of any series of transformations starting from frame i and ending in frame j should be identical to $\mathbf{M}_{ij}$ (See Fig. 1). Due to the presence of noise in our observations the various transformation estimates would not be consistent with each other. Hence $\mathbf{M}_{ij} \neq \mathbf{M}_j {\mathbf{M}_i}^{-1}$, where $\mathbf{M}_{ij}$ is the estimated transformation between frames i and j. However we can rewrite the given relationship as a constraint on the global motion model $\{\mathbf{M}_2, \cdots, \mathbf{M}_N\}$ which completely describes the motion[3]. Since in general we have upto $\frac{N(N-1)}{2}$ such constraints, we have an over-determined system of equations.

$$\mathbf{M}_j {\mathbf{M}_i}^{-1} = \mathbf{M}_{ij}, \forall i \neq j \tag{8}$$

where the variables on the left-side are unknowns to be estimated ("fitted") in terms of the observed data $\mathbf{M}_{ij}$ on the right. Intuitively, we want to estimate a global motion model $\{\mathbf{M}_i\}$ that is most consistent with the measurements $\{\mathbf{M}_{ij}\}$ derived from the data. Thus the errors in individual estimates of $\mathbf{M}_{ij}$ are "averaged" out which is very useful for long and/or closed loops where usually the solution drifts off due to accumulated error. Consider the scenario where the last frame in a sequence is close to the first frame. In a conventional method, the errors between adjacent frames would accumulate causing large errors. But in our framework, we can incorporate an estimate of the relative motion between the last and the first frame, resulting in a constraint that would "close" the loop and cause the errors to be reduced and also be more evenly distributed along the sequence.

It may be noted that in Eqn. 8, we are not required to use every pairwise constraint to get a solution. For extended sequences, there is seldom any overlap between frames well separated in time, therefore their relative two-frame motions cannot be estimated. However we can still get a consistent solution as long as we have at least $N-1$ relative motions.

3.1. Lie Averaging of Relative Motions

The scheme for averaging relative motions is similar in spirit to the intrinsic averaging algorithm (Algorithm **A1**) described in Sec. 2.1. We recall that $\mathbf{M}_{ij}$ and therefore $\mathfrak{m}_{ij}$ are the relative motions estimated from the image data, and $\mathbf{M}_k$ and in turn $\mathfrak{m}_k$ are the global motion models we want to estimate. Here $\mathbf{M}_k$ denotes the motion of frame k with respect to the first frame. Starting from the constraint $\mathbf{M}_{ij} = \mathbf{M}_j {\mathbf{M}_i}^{-1}$, by applying the first-order approximation to the Riemannian distance given in Eqn. 4, we have $\mathbf{M}_{ij} = \mathbf{M}_j {\mathbf{M}_i}^{-1} \Rightarrow \mathfrak{m}_{ij} = \mathfrak{m}_j - \mathfrak{m}_i$ since $\mathfrak{m} = \log(\mathbf{M})$. Now the matrices $\mathfrak{m}$ can be described by 3 or 6 parameters for $\mathfrak{m} \in \mathfrak{so}(3)$ and $\mathfrak{m} \in \mathfrak{se}(3)$ respectively. If we arrange these parameters in the form of a column vector, say $\mathfrak{v} = vec(\mathfrak{m})$ (where $vec(.)$ returns a column of parameters extracted from the input matrix), the same relationship holds, i.e. $\mathfrak{v}_{ij} = \mathfrak{v}_j - \mathfrak{v}_i$. If we stack all the column vectors for the global motion model into one big vector $\mathfrak{V}$ we have $\mathfrak{V} = [\mathfrak{v}_2; \cdots; \mathfrak{v}_N]$. Given this unified vector representation for the global motion model, we have the following relationship

$$\begin{aligned} \mathbf{M}_{ij} &= \mathbf{M}_j {\mathbf{M}_i}^{-1} \Rightarrow \mathfrak{m}_{ij} = \mathfrak{m}_j - \mathfrak{m}_i \\ \Rightarrow \mathfrak{v}_{ij} &= \mathfrak{v}_j - \mathfrak{v}_i \\ \Rightarrow \mathfrak{v}_{ij} &= \underbrace{\begin{bmatrix} \cdots -\mathbf{I} \cdots \mathbf{I} \cdots \end{bmatrix}}_{=\mathbf{D}_{ij}} \mathfrak{V} \end{aligned} \tag{9}$$

where $\mathbf{I}$ denotes an identity matrix of dimensions $N_{dim} \times N_{dim}$ (where N_{dim} = 3 or 6). $\mathbf{D}_{ij}$ is a matrix of size $N_{dim} \times (N_{dim} \times N - 1)$ with matrices $-\mathbf{I}$ and $\mathbf{I}$ at positions i and j respectively and acts as an "indicator" matrix for the frames i and j. In other words, applying $\mathbf{D}_{ij}$ to the vector $\mathfrak{V}$ picks out $\mathfrak{v}_i$ and $\mathfrak{v}_j$ and returns $\mathfrak{v}_j - \mathfrak{v}_i$. Now Eqn. 9 denotes a single relative motion in terms of the global motion model. However we need to combine all the available relative motions into a single set of equations. Thus, for

[3]The first image being the reference frame, $\mathbf{M}_1$ is an identity transformation

a given set of relative motion observations $\{\mathbf{M}_{ij}\}$, we can stack all the relative motion vectors $\mathfrak{v}_{ij}$ into one big vector $\mathbb{V}_{ij} = [\mathfrak{v}_{ij1}; \mathfrak{v}_{ij2}; \cdots]$ where $ij1, ij2$ etc. denote different relative motion indices. Similarly all the indicator matrices can be stacked into one big matrix $\mathbf{D} = [\mathbf{D}_{ij1}; \mathbf{D}_{ij2}; \cdots]$ leading to the following unified representation

$$\begin{aligned} \mathbf{M}_j {\mathbf{M}_i}^{-1} &= \mathbf{M}_{ij} \\ \rightsquigarrow \mathbf{D}\mathfrak{V} &= \mathbb{V}_{ij} \\ \Rightarrow \mathfrak{V} &= \mathbf{D}^{\dagger}\mathbb{V}_{ij} \end{aligned} \tag{10}$$

where $\mathbf{D}^{\dagger}$ represents the pseudo-inverse operation, $\mathbf{D}^{\dagger} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T$. It should be noted that for a given set of relative observations, the matrix $\mathbf{D}$ is fixed implying that $\mathbf{D}^{\dagger}$ needs to be computed only once.

Now similar to the scheme for averaging group elements in Sec. 2.1 one can develop an iterative scheme whereby at each step, we linearly approximate the global motion and update the current estimate from the observed motion values to improve the estimate. This can be repeated till convergence resulting in the following algorithm

A2 : Algorithm for Motion Averaging

Input : $\{\mathbf{M}_{ij1}, \mathbf{M}_{ij2} \cdots, \mathbf{M}_{ijn}\}$ (n relative motions)
Output : $\mathbf{M}_g : \{\mathbf{M}_2, \cdots, \mathbf{M}_N\}$ (N image global motion)
Set $\mathbf{M}_g$ to an initial guess (Linear solution in [7])
Do
$\Delta\mathbf{M}_{ij} = {\mathbf{M}_j}^{-1}\mathbf{M}_{ij}\mathbf{M}_i$
$\Delta\mathfrak{m}_{ij} = log(\Delta\mathbf{M}_{ij})$
$\Delta\mathfrak{v}_{ij} = vec(\mathfrak{m}_{ij})$
$\Delta\mathfrak{V} = \mathbf{D}^{\dagger}\Delta\mathbb{V}_{ij}$
$\forall k \in [2, N], \mathbf{M}_k = \mathbf{M}_k exp(\Delta\mathfrak{v}_k)$
$Repeat\ till\ ||\Delta\mathfrak{V}|| < \epsilon$

In the above algorithm, for simplicity of notation we have omitted a few steps which bear mentioning here. The vector $\Delta\mathbb{V}_{ij}$ is created by stacking all the relative motion vectors $\Delta\mathfrak{v}_{ij}$ and conversely the vectors $\Delta\mathfrak{v}_k$ are extracted out of the linear estimate $\Delta\mathfrak{V}$. Also a crucial point to observe is that our estimation of the residual motions has the specific form $\Delta\mathbf{M}_{ij} = {\mathbf{M}_j}^{-1}\mathbf{M}_{ij}\mathbf{M}_i$ so as to respect the non-commutative nature of matrix multiplication. The astute reader would also notice that a single iteration of this algorithm would amount to computing the *extrinsic* average of the relative motions. However like in the previous algorithm (**A1**) here we improve this average by means of iteratively improving the quality of the first-order approximation made in using the Lie algebra elements $\mathfrak{m}$ to compute the averages. Empirically one can note that this algorithm converges to a stable point within a few iterations. Depending on the length of the image sequence (i.e. N ranging from 5 to 20), the method converges in $2-5$ iterations in our experiments. The computational time is typically a fraction of a second for these experiments. It should be noted that the number of iterations required would also depend on the amount of noise in our observations since highly noisy image data would imply that the relative motions would have large scatter and it would require longer for the estimated average to converge.

Two important issues (i.e. robustness and weighted estimation) not dealt with in this paper can be easily incorporated into both the algorithms explained above. The quality of our solution can be used to identify input elements ($\mathbf{M}_{ij}$'s) that are erroneous and need to be eliminated from the computation. Also in the case where relative weights are available for individual observations, we can directly incorporate them into the algorithm to reflect our confidence in each individual $\mathbf{M}_{ij}$. For example, in algorithm **A2**, multiplying both sides of the last equation in Eqn. 9 by a given weight would result in a weighted average estimate.

Noise	Lie-Average		Bundle Adjustment	
	Error	Flops ($\times 10^6$)	Error	Flops ($\times 10^6$)
0.5	0.22	1.40	0.10	90
1.0	0.50	1.41	0.17	95
1.5	0.75	1.42	0.25	109
2.0	0.97	1.43	0.35	126
2.5	1.19	1.43	0.43	138

Table 1: Results for 20 points and 5 images averaged over 50 trials; Noise Level is in pixels for 256×256 image

Noise	Lie-Average		Bundle Adjustment	
	Error	Flops ($\times 10^6$)	Error	Flops ($\times 10^6$)
0.5	0.20	1.55	0.10	853
1.0	0.47	1.56	0.17	908
1.5	0.61	1.57	0.20	996
2.0	0.77	1.57	0.27	1058
2.5	0.91	1.56	0.30	1122

Table 2: Results for 50 points and 5 images averaged over 50 trials; Noise Level is in pixels for 256×256 image

4. Experimental Results

In this Section we describe a comparison of the Lie-averaging scheme with bundle-adjustment and demonstrate the results of motion estimation using our method on two

real image sequences.

The results of a MATLAB implemented empirical comparison of our algorithm with the optimal reprojection method that uses bundle-adjustment are summarised in Tables 1 and 2. Feature points were generated for 5 images (within a 60° field-of-view with random motions) and subjected to varying levels of Gaussian noise (in pixels for an equivalent image size of 256×256 pixels). For the Lie-averaging algorithm, the inputs of relative motions were computed using the eight-point algorithm [8]. It may be recalled that the eight-point algorithm does not provide the scale of the translation. We choose to fix these scales by using a linear-fit of the geometries as described in [7]. In the case of bundle-adjustment, we provide the algorithm with our Lie-average motion estimate and ground-truth structure as the initial value. The errors shown are the L_2 norm of the difference in motion estimates for the sequence, i.e. for ground truth motion $\mathbf{M}_g$ and estimated motion $\mathbf{M}_e$, the error is the L_2 norm for $vec(log(\mathbf{M}_g{\mathbf{M}_e}^{-1}))$. We prefer this measure over the usual method of comparing reprojection error since here we are interested in only recovering the camera motion and our approach is independent of the object structure. The results summarised are averages over 50 trials and demonstrate that our approach gives reasonable results at a fraction of the computational cost of bundle-adjustment. This becomes more pronounced with larger number of correspondences since bundle-adjustment has to minimise over the structure variables whereas our averaging method is independent of the structure (except for the marginal cost of the eight-point algorithm). Thus we can see that our method is in general significantly faster than bundle-adjustment based motion estimation and can serve as a good initialisation in the case where a global bundle-adjustment over structure and motion is preferred.

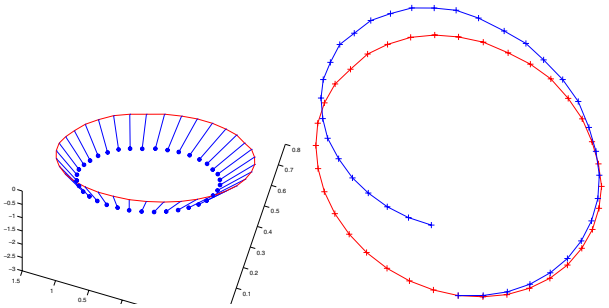
The first of our real image experiments is a sequence of 36 images of a toy dinosaur placed on a turntable ([4]). The 36 images correspond to images taken 10° apart (covering one revolution of the turntable) with a fixed axis of rotation. The special single-axis geometry of turntables has been exploited to build high-quality 3D models ([4], [11]). In our algorithm we do not explicitly use this information of the sequence but treat it as a general sequence. However given the high redundancy of relative motion measurements we can acquire from the sequence we can estimate the camera motion for this sequence to a high degree of accuracy.

Using features tracked over the sequence, we estimate the epipolar geometry of 231 image pairs. The epipolar geometry is estimated using the eight-point algorithm [8]. We calibrate the sequence by using a simple technique for focal length estimation as given in [15]. However, to improve the accuracy of the geometry estimates, we wish to get more feature correspondences between image pairs. This is achieved in the following manner by using epipolar geometry to guide the selection of more correspondences. A corner detector is run on all images and all correspondences between images i and j that confirm closely to the epipolar geometry of F_{ij} are selected [4]. The epipolar geometry between images i and j is now updated using these newly estimated feature correspondences, thereby increasing the accuracy of the estimates since the guided matching provides us many more correspondences.

An estimate of the global motion of the image sequence is obtained using the algorithm for Lie-averaging described in Sec. 3.1 (Algorithm **A2**). Since the sequence is closed, we also have relative motion estimates between images towards the end of the sequences and those at the beginning of the sequence. As a result, our averaging scheme implicitly enforces global consistency on the motion estimate. As a baseline for comparison, we also chose to estimate the motion of the sequence by cascading motion estimates between adjacent images, i.e. between image pairs $(1,2),(2,3),\cdots(35,36)$. The results are shown in Fig. 2. As can be seen in Fig. 2(b) in the baseline estimate, the translation errors accumulate resulting in significant drift of the estimate (open loop). On the other hand the translation estimate using our Lie-averaging scheme is highly accurate and the circular loop is closed. This is so because the epipolar geometry between the images at the end of the sequence and at the beginning of the sequence provides enough constraints for the loop to be closed. Similarly the rotation estimate can also be seen to be highly accurate. In fact, the ground truth for this data set is given as rotation in steps of 10° with a standard deviation of 0.05°. Using our Lie-averaging motion estimation method, we obtain a rotation average of 10° with a standard deviation of 0.1° which compares quite well with the ground truth.

An incidental aspect of this processing is that since we do not solve for structure anywhere in the estimation, each relative motion estimate has an unknown translation scale, i.e. epipolar geometry can only solve for translation direction. However since we are averaging elements of $\mathbf{SE}(3)$, we need to fix the scale which is accomplished in the following manner. During each iteration of our averaging algorithm, we use the current estimate of motions to fix the scale of the corresponding translation measurement, i.e. the scale of translation for $\mathbf{M}_{ij}$ is fixed to be equal to that of $\mathbf{M}_j{\mathbf{M}_i}^{-1}$ where $\mathbf{M}_i$ and $\mathbf{M}_j$ are the "current" estimates of camera motions. These scales are updated for each iteration till

[4] Since the motion is uniform, we use all correspondences between images that are d images apart to estimate F_{ij} (for $|i-j|=d$). But these estimates are only used for guided matching and are thereafter discarded.



(a) Recovered Motion (b) Translation Comparison

Figure 2: (a) shows the estimated motion with the viewing direction of each camera indicated (not to scale). (b) shows the accuracy of the translation estimate in comparison with the drift in the baseline estimate.

convergence [5]. Also the algorithm is initialised with an estimate obtained using the linear method described in [7].

In our second experiment, we show the result of applying our averaging technique to the Castle sequence used in [16, 6] (Fig. 3(a)) of which 23 images were used. The focal length of the sequence was estimated using the calibration technique of [15] [6]. We use the gradient-based epipolar geometry method in [22] to derive the estimates of relative motion between images of this sequence. As a preprocessing stage, using the method in [7] we linearly solve for camera motion for triplets of adjacent images and measure the accuracy of fit. The accuracy of this fit is used to eliminate frames of poor quality, i.e. those frames that are either too close to other frames or ones that have bad epipolar geometry estimates. If required these frames could be incorporated later into the sequence using camera resectioning but is not carried out in this work. Subsequent to this frame elimination there is no further preprocessing involved. All the reliable estimates of relative motion are averaged using the Lie-averaging method to solve for the global motion.

The camera trajectory with the camera orientation superimposed is shown in Fig. 3(b). The trajectory is shown with the viewing direction of each camera indicated. The length of the line indicates the magnitude of rotation of the frame with respect to the reference frame. Although we do not have any ground truth information for this sequence, a comparison with the result presented in [6] suggests that this result is reasonably accurate. Any subsequent refinement of the solution using bundle-adjustment will be relatively easier due to the good initialisation provided by our result.

[5] Note that if one were using a known 3D object to recover pose or if we reconstruct a few points over the sequence, one could use the knowledge of depth to fix the scales. But we choose to avoid computing the structure terms at any stage in the algorithm.

[6] The images used here are of size 384×288 and the focal length estimate is about 600 pixels, i.e. the images used here are half the original size used in [16] which should affect the quality of our result to some extent.

5. Conclusions

In this paper, we have presented a method of averaging relative motion estimates using a Lie-algebraic framework. The method presented has the flexibility of linearly computing all possible relative motions and averaging these estimates in a fast manner. The results presented demonstrate the accuracy that can be achieved using such a Lie averaging algorithm. One obvious use of the Lie averaging idea is for fusing information derived from different sensors. Another area of application is robotic path estimation where often the error drift needs to be corrected in the case of long sequences that may have closed loops. In fact, the averaging method outlined here is similar in spirit to [13] and is a well-motivated method of globally consistent motion averaging.

6. Acknowledgments

Andrew Fitzgibbon and Andrew Zisserman of Oxford University provided the dinosaur sequence data and thanks are also due to Efrat Egozi and Michael Werman of HUJI for the castle sequence correspondences.

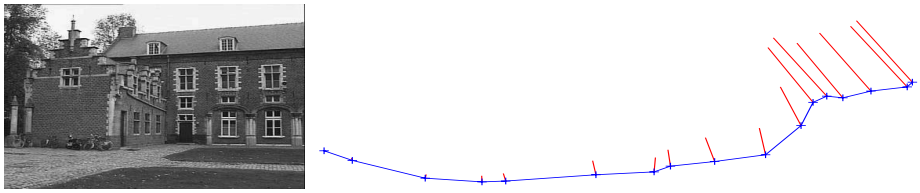
References

[1] Buss, S. R. and Fillmore, J.P., "Spherical averages and applications to spherical splines and interpolation", *ACM Transactions on Graphics*, vol. 20(2): pp. 95-126, 2001.

[2] Curtis, W.D., Janin, A. L., and Zikan, K., "A Note on Averaging Rotations", *IEEE Virtual Reality Annual International Symposium*, pp. 377-385, 1993.

[3] Engo, K., "On the BCH formula in $\mathfrak{so}(3)$", *Department of Informatics, University of Bergen*, Report No. 201, September 2000.

[4] Fitzgibbon, A., Cross, G., and Zisserman, A., "Automatic 3D Model Construction for Turn-Table Sequences", *Structure and Motion from Multiple Images in Large-Scale Environments*, pp. 154-169, Lecture Notes in Computer Science, Springer-Verlag, 1998.



(a) Castle Sequence (b) Estimated Motion

Figure 3: (a) shows a frame from the Castle sequence. (b) shows the estimated motion for the sequence viewed along the y-axis. The curve indicates the trajectory of the camera. The lines at individual camera positions indicate the viewing direction of the camera and the length of the line is proportional to the magnitude of the rotation angle with respect to the first frame.

[5] Fletcher, P. T., Lu, C., and Joshi, S., "Statistics of Shape via Principal Component Analysis on Lie Groups", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 95-101, 2003.

[6] Georgescu, B. and Meer, P., "Balanced Recovery of 3D Structure and Camera Motion from Uncalibrated Image Sequences", *European Conference on Computer Vision*, pp. 294-308, 2002.

[7] Govindu, V. M., "Combining Two-view Constraints for Motion Estimation", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 218-225, 2001.

[8] Hartley, R., "In Defence of the 8-point algorithm", *Proceedings of the 5th International Conference on Computer Vision*, IEEE Computer Society Press, pp. 1064-1070, 1995.

[9] Hartley, R., "Euclidean Reconstruction from Uncalibrated Views", *Proceedings of the DARPA-ESPIRIT Workshop on Applications of Invariance in Computer Vision*, pp. 187-202, 1993.

[10] Hartley, R. and Zisserman, A., *Multiple View Geometry in Computer Vision,* Cambridge University Press, 2000.

[11] Jiang, G., Tsui, H.-t., Quan, L., and Zisserman, A., "Single Axis Geometry by Fitting Conics", *Proceedings of the 7th European Conference on Computer Vision*, pp. I : 537 ff, 2002.

[12] Kanatani, K., *Group-Theoretical Methods in Image Understanding,* Springer-Verlag, 1990.

[13] Lu, F., and Milios, E., "Globally Consistent Range-Scan Alignment for Environment Mapping", *Autonomous Robots*, pp. 333-349, 1997.

[14] Moakher, M., "Means and averaging in the group of rotations", *SIAM Journal on Matrix Analysis and Applications*, vol. 24(1), pp. 1-16, 2002.

[15] Mendonca, P. R. S., and Cipolla, R., "A Simple Technique for Self-Calibration", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 112-116, 1999.

[16] Pollefeys, M., "Self-calibration and Metric Reconstruction in spite of Varying and Unknown Intrinsic Camera Parameters", *International Journal of Computer Vision*, vol 32, pp. 7-25, 1999.

[17] Selig, J.M., *Geometrical Methods in Robotics*, Springer-Verlag, 1996.

[18] Sturm P., and Triggs, B., "A Factorization Based Algorithm for Multi-Frame Projective Structure and Motion", *Proceedings of the 4th European Conference on Computer Vision*, pp. 709-720, 1996.

[19] Szeliski R. and Kang S. B., "Recovering 3D Shape and Motion from Image Streams using Nonlinear Least Squares," *Journal of Visual Communication and Image Representation*, vol. 5, pp. 10–28, 1994.

[20] Tomasi, C. and Kanade, T., "Shape and Motion from image streams under orthography : A factorization method," *International Journal of Computer Vision*, vol. 9(2), pp. 137-154, 1992.

[21] Varadarajan, V.S. *Lie Groups, Lie Algebras and Their Representations*, Springer-Verlag. Graduate Texts in Mathematics, vol. 102, 1984.

[22] Zhang Z., "Determining the Epipolar Geometry and its Uncertainty: A Review", *INRIA Research Report No. 2927*, July 1996.