Non-Iterative, Fast SE(3) Path Smoothing

Yonhon Ng[1], Bomin Jiang[2], Changbin Yu[1] and Hongdong Li[1,3]

Abstract— **In this paper, we present a fast, non-iterative approach to smooth a noisy input on the Special Euclidean Group, SE(3) manifold. The translational part can be smoothed by a simple Gaussian convolution. We then proposed a novel approach to rotation smoothing. Unlike existing rotation smoothing methods using either iterative optimization methods or stochastic filtering methods, our method allows direct computation of the smoothing result and allows parallelization of the computation. Furthermore, we have done a comparative study on Jia and Evans's method published in 2014 [1], and shown that our method can better smooth an input rotation sequence, with shorter computational time. The smoothed camera path is then used for video stabilisation, which shows fluid and smooth camera motion.**

I. INTRODUCTION

In recent years, cameras are becoming a staple sensor on a robotics platform. This is due to the wide variety of information that can be obtained purely from a sequence of video captured by an onboard camera. For example, for teleoperation [2][3][4], estimate position and scene reconstruction [5], or surveillance [6].

As a result, the problem of video enhancement, e.g. contrast enhancement, [7], encoding and compressing [8], video stabilization [9], etc. has been studied with increasing intensity. In many scenarios, such video enhancement techniques are required to be implemented in real time [10] with minimum computation [7] while sustaining high quality result.

Our paper studies one of these problems, the video stabilization problem, via rotation smoothing. The common way to smooth a noisy Riemannian manifold is by specifying a cost function, and using iterative method like gradient decent [11], Newton's method [12], Lagrangian Duality [13], or Iteratively Reweighted Least Square [14], [15] to minimise the cost function. On the other hand, there are also works based on stochastic filtering methods [10], [16].

However, there are a number of drawbacks in iterative cost reduction or optimisation. Iterative methods tend to require a suitable initialisation for convergence, and the number of iterations before convergence is not fixed for different input. Thus, iterative methods may not guarantee a global convergence, and can be problematic for real-time implementation.

In addition, the implementation of stochastic filtering method requires an unbiased prior, which is not always available in practice. Also, due to the lack of actual future measurements, the output of stochastic filtering methods may produce a result that deviates from the mean of the actual motion more than methods that uses that extra information.

Unlike existing rotation smoothing methods using either iterative optimization methods [1] or stochastic filtering methods [10], [16], our method allows direct computation of the smoothing result and allows parallelization of the computation. As shown by a comparative study in our paper, our proposed method achieves smoother result and shorter computational time in comparison to the one proposed by Jia and Evans [1].

It is also noted that the application of our rotation smoothing method is not limited to video stabilization. In a robotics system, there is a need to measure physical properties using sensors, which are bound to have noise in their measurements. There may also be actual perturbation in the physical property being measured, which may mask out the actual intended motion. The need to smooth pose (position and orientation) data from the IMU is a commonly encountered problem in the field of mobile robotics.

II. BACKGROUND

In signal processing community, there is a well known and widely used method to smooth a vector (e.g. $\mathbb{R}^3$) signal, that is, by convolving the noisy signal with a Gaussian kernel.

This is useful in smoothing the translational component, $\boldsymbol{t}$, in the $\mathbf{SE(3)}$ space. This can be done by a simple weighted average of arithmetic mean , such that, for 3D translations, $\boldsymbol{t}_i \in \mathbb{R}^3$, the cost minimising function can be defined as, $F(\boldsymbol{t}_i) := \sum_j g_j.\boldsymbol{t}_{i+j}$. This equation can also be seen as a convolution.

The other component of $\mathbf{SE(3)}$ group is the 3D rotation. 3D rotation lies in the Special Orthogonal Group, $R \in \mathbf{SO(3)}$, which requires special handling. The main focus of this paper will be on proposing a fast, non-iterative approach to smooth an input rotation sequence.

Two rotations can be averaged with different weighing factor. The method to calculate the weighted average of two rotation matrices is discussed in [17].

An alternative way is to use rotations in quaternion form in finding the weighted average of two rotations, using a method called *slerp* [18]. *Slerp* relies on linear interpolation on non-unique quaternion representation (**q=-q**), which suffers from rotation direction changing abruptly.

[1]Yonhon Ng, Changbin (Brad) Yu and Hongdong Li are with Research School of Engineering, Australian National University, Acton ACT 2601, Australia (email: {yonhon.ng, brad.yu, hongdong.li}@anu.edu.au).

[2]Bomin Jiang is with Laboratory for Information & Decision Systems, and the Department of Mechanical Engineering, Massachusetts Institution of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA (email: bominj@mit.edu)

[3]Hongdong Li is also with the Australian ARC Centre of Excellence for Robotic Vision (CE140100016), Australian National University, Acton ACT 2601, Australia

This means that the weighted average found using *slerp* may not always lie on the shortest geodesic curve between the two rotations. Thus, the weighted average using rotation matrices produces more stable result.

On the other hand, an iterative algorithm to compute the Geodesic L_2 mean of multiple rotational matrices was presented in [11], [19]. The rotation minimizing cost function is, $C(R) = \sum_{i=1}^{n} d_{\angle}(R, R_i)^2$. Geodesic L_2 Mean is also known as the Karcher mean of rotations, or the geometric mean [17].

In addition, Geodesic L_q Mean can be found by using Iteratively Reweighted Least Square (IRLS) Method. IRLS is also called L_q Weiszfeld Algorithm [14]. It is a method that minimises the cost function, $C_q(R) = \sum_{i=1}^{n} d_{\angle}(R, R_i)^q$.

As the name suggests, it relies on the iterative Geodesic L_2 Mean algorithm by adding an extra reweighing factor, $w_i = d_{\angle}(R, R_i)^{q-2}$. This reweighing for different R and R_i pair changes the gradient decent Geodesic L_2 Mean method to solve for Geodesic L_q Mean instead.

The distance, $d_{\angle}(R, R_i) := (1/\sqrt{2})||log(R^{-1}R_i)||_F$, where $|| * ||_F$ is the Frobenius norm of the matrix, and the scaling $(1/\sqrt{2})$ ensures that $d_{\angle}(*, *)$ represents the angular distance between the two rotations.

When $q = 1$, the algorithm finds the Geodesic L_1 Mean solution, which is known to be more robust against outliers in the input. However, Geodesic L_1 mean computation is also known to be slower than Geodesic L_2 mean, and there is also a possibility of the solution getting stuck when R is equals to one of the input, $R_j \in R_i$. [14]

When R is equals to one of the input R_j, the weighing factor, w_j (in Algorithm 3) becomes $\frac{1}{0} = \infty$. Thus, all the other weighing factors will be insignificant compared to the weighing factor of R_j, and the small change in rotation, $\mathbf{r}$ will become the identity matrix (solution is stuck).

We can partially overcome the slow convergence by choosing $1 < q < 2$, as discussed by [14]. However, this does not solve the problem of solution getting stuck when R is equals to R_j, because w_j still approaches ∞ as R approaches R_j, albeit at a slower rate.

Like all iterative algorithm, there needs to be a good initial estimate of R. As suggested by Aftab et al. [14], the initial estimate can be found by Chordal L_2 Mean, which has a closed-form solution.

Chordal L_2 Mean is defined as the rotation which minimises the cost, $C(R) = \sum_{i=1}^{n} d_{chord}(R, R_i)^2$. It is also named the projected or induced arithmetic mean. [17], [20]

The algorithm to compute Chordal L_2 Mean is given by [19], which uses Singular Value Decomposition (SVD) instead of polar decomposition used in [17], [20]. Reprojecting the algebraic sum of the rotational matrices onto the orthogonal $\mathbf{SO(3)}$ manifold is also called the Orthogonal Procusthes Problem. [21] [22]

Another way to smooth an input sequence of Rotation (Orientation) is to specify a cost function to minimise. Jia and Evans [1] proposed a similar cost function as the equation shown as follows.

$$\min_{\{R_i\}} \sum_{i=1}^{N} d(\tilde{R}_i, R_i) + \alpha \sum_{i=1}^{N-1} d(R_i, R_{i+1}) \tag{1}$$

Where,

$d(*, *)$ is any suitable distance metric in $\mathbf{SO(3)}$. (eg. Geodesic L_2, Geodesic L_1, Chordal L_2, Chordal L_1, Quaternion L_1, Quaternion L_2, etc.) More information of different distance metric can be found in [19].

α is the scalar factor controlling the smoothness of the output trajectory (a trade-off against deviation from input - the first summation term in the cost function)

$\tilde{R}_i$ is the input (measured) orientation at the i^{th} instance

R_i is the smoothed orientation at the i^{th} instance

III. PROPOSED METHOD

In this section, we propose some new methods to do weighted average of rotations. These methods can be combined with the Gaussian smoothing technique to smooth a sequence of rotational matrix.

A. Pairwise Gaussian Weighted Average of 2^n Vectors

A Gaussian filter kernel can be calculated using the equation as follows.

$$G(t|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} \tag{2}$$

For a discrete case, to ensure that the resulting filtered signal has the same scale as the original, we can make sure that the sum of all the elements of the kernel is 1 by a simple normalisation.

$$G_{norm}(t|\mu, \sigma^2) = \frac{G(t|\mu, \sigma^2)}{\sum(G(t|\mu, \sigma^2))} \tag{3}$$

The Gaussian Kernel values are calculated by using t with equal spacing, centred around zero ($\mu = 0$). For temporally invariant Gaussian Kernel, the value of t is the time at each measurements, and μ is the time of the middle entry of the input (mean of Gaussian distribution).

We then introduce a method to rearrange a normalised weighted sum of two vectors, into a form similar to the equation of weighted average of two rotations (5).

A normalised weighted sum of two vectors $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ with different weighing factor (g_1, g_2) is equivalent to the difference in value $\boldsymbol{x}_2 - \boldsymbol{x}_1$ multiplied by the ratio of the weighing factor of $\boldsymbol{x}_2$ to the total weighing factor, added to $\boldsymbol{x}_1$. The result is included as follows.

$$\begin{aligned} \frac{1}{g_1 + g_2}(g_1\boldsymbol{x}_1 + g_2\boldsymbol{x}_2) &= \boldsymbol{x}_1 + \frac{g_2}{g_1 + g_2}(\boldsymbol{x}_2 - \boldsymbol{x}_1) \\ &= \boldsymbol{x}_1 + \lambda(-\boldsymbol{x}_1 + \boldsymbol{x}_2) \end{aligned} \tag{4}$$

We propose that the weighted average of 2^n vectors can be decomposed into a sequence of pairwise averages. This will be useful in the next Section. The operation can be illustrated as shown in Fig. 1.

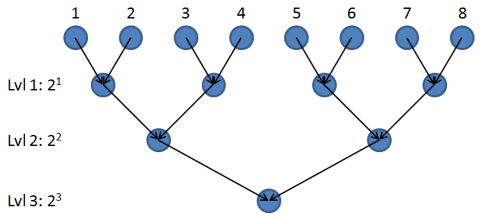


Fig. 1: 2^n Averaging Tree

Each arrow in Fig. 1 shows a normalised weighted average operation between two vectors. The updated weighing factor of each average operation is the sum of the corresponding weighing factors of that average.

This method is equivalent to a weighted averaging filter, with window size of 2^n, which is capable of smoothing an otherwise noisy input vector (e.g. translational component of $\mathbf{SE(3)}$).

B. *Gaussian Weighted Average of 2^n Rotations*

The weighted average of two rotations, which lies on the shortest geodesic curve connecting the two rotations can be calculated as follows.

$$R_{weightedAve} = R_1\ exp(\lambda\ log(R_1^T R_2)) \quad (5)$$

where, $log(*)$ is the matrix logarithm, and $exp(*)$ is the matrix exponential. The exponential and logarithm of the rotation group are also discussed in Moakher's paper [17].

Note that the matrix logarithm and exponential method shown above cannot be used directly on the $\mathbf{SE(3)}$ manifold, because unlike $\mathbf{SO(3)}$, the result of the operation cannot be guaranteed to remain on that manifold.

In the rest of this section, we propose a novel, generalised method to perform weighted average of multiple rotations deterministically.

In order to find a weighted average of rotations in a window size of 2^n, a method similar to that presented in Section III-A may be used. Instead of vectors, each circle (node) represents a rotation.

This is well defined because the weighted average of two input rotations can be calculated exactly using (5).

Similar to weighted average of 2^n vectors, only the ratio of their corresponding weight matters. After each pairwise average, their resulting weighing factor is equivalent to the sum of their corresponding weights. Fig. 2 illustrates this concept with an example.

With this generalised method in finding the weighted average of 2^n rotations, we can then do Gaussian filtering in a similar way to the vector case (Section III-A).

Although in the case of averaging rotations, Bingham Distribution is more appropriate due to the wrap around effect, but as discussed in Kurz et al. [23], it was shown that for standard deviation less than $11°$, Gaussian Distribution is a good approximation.

It is also noted that there is a need to account for the delay introduced by the filter, which is equivalent to $\frac{2^n+1}{2}$, as can be seen from the graph in Fig. 1. The resulting average is

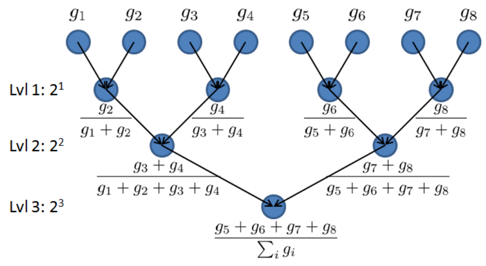


Fig. 2: 2^n Weighted Averaging Tree, with their weight, λ in (5) shown below the nodes

thus a value for rotation between the forth and fifth values used.

In order to reposition the averaged value to align to an input time interval, we can do another average between subsequent averaged value as shown in Fig. 3. This is equivalent to an interpolation step of the two consecutive rotation averages.

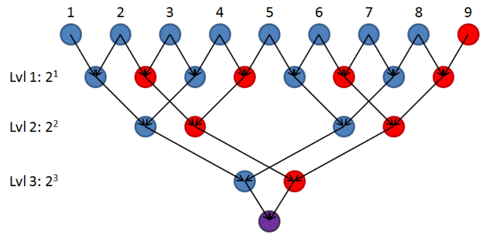


Fig. 3: 2^n Averaging Tree with Value Reposition

The red circles are the values used and computed for one time step after the blue circles, and the purple value is the repositioned average (which is aligned to the fifth input value).

By combining the proposed pairwise weighted average of 2^n rotations and the Gaussian Filter technique, we have found a way to smooth a sequence of 3D rotation data.

It is noted that every layer (or level) contains completely independent computations of pairwise rotation averaging. Thus, they are parallelisable for faster computation. For example, Fig. 3 shows window size of 9, and there are a total of 15 pairwise averages, but after parallelisation, only 4 dependent levels are left (a potential for $3.75\times$ shorter computation time).

We can summarise the pairwise method in Algorithm 1 as follows.

C. *Gaussian Weighted Geodesic L_2 Mean*

We have also explored the possibility of using the Gaussian weighing factors, g_i when doing the iterative Geodesic L_2 distance minimisation. In essence, we just modify the 3^{rd} line of Algorithm 2 from [19] as follows.

D. *Gaussian Weighted Geodesic L_q Mean*

Similar to Weighted Geodesic L_2 Mean, we can add an extra weighing factor, g_i to the Geodesic L_q Mean [14]. The resulting algorithm is given in Algorithm 3.

Algorithm 1 Gaussian Weighted Pairwise Average on **SO(3)**

```
1:  loop
2:      parallel loop (Level 1)
3:        Compute $R_{lvl1,i} = R_{2i-1} exp(g_i\ log(R^T_{2i-1} R_{2i}))$
4:        where $i \in [1, N]$, and $N$ is half the window size
5:      end parallel loop
6:      parallel loop (Level 2)
7:        Compute
8:         $R_{lvl2,i} = R_{lvl1,2i-1} exp(\lambda_i\ log(R^T_{lvl1,2i-1} R_{lvl1,2i}))$
9:         where $\lambda$ is the updated weighing factor (Fig. 2)
10:     end parallel loop
11:     ... (more parallel loops until the last two averages)
12: end loop
```

Algorithm 2 Gaussian Weighted Geodesic L_2 Mean on **SO(3)**

```
1: Set $R := R_{mid}$. Choose a tolerance $\epsilon > 0$
2: loop
3:     Compute $\mathbf{r} := \sum_{i=1}^{n} g_i\ log(R^T R_i)$
4:     if $||\mathbf{r}|| < \epsilon$ then
5:         return R
6:     end if
7:     Update $R := R\ exp(\mathbf{r})$
8: end loop
```

E. Gaussian Weighted Chordal L_2 Mean

Instead of a simple algebraic sum in the original Chordal L_2 Mean [19], we can do a weighted sum instead. The proposed modification to C_e is as shown in Line 1 of Algorithm 4.

The addition of Gaussian weighs to the rotation averaging methods makes the algorithms more robust against averaging large angular motion, because the weight given to the middle entry of the window is higher than those further to the edges. Thus, preserving the continuity of the smoothed motion.

The results using our Pairwise Average method, Jia and Evan's method, the Gaussian Weighted Geodesic L_2 Mean method, and other window-based methods discussed are presented and compared in Section IV.

Algorithm 3 Gaussian Weighted Geodesic L_q Mean on **SO(3)**

```
1:  Set $R := R_{initial}$. Choose a tolerance $\epsilon > 0$
2:  loop
3:      Compute
4:          $\mathbf{r} := \left(\sum_{i=1}^{n} g_i\ w_i\ log(R^T R_i)\right) / (\sum_{i=1}^{n} w_i)$,
5:          where, $w_i = (d_\angle(R, R_i))^{q-2}$
6:      if $||\mathbf{r}|| < \epsilon$ then
7:          return R
8:      end if
9:      Update $R := R\ exp(\mathbf{r})$
10: end loop
```

Algorithm 4 Gaussian Weighted Chordal L_2 Mean on **SO(3)**

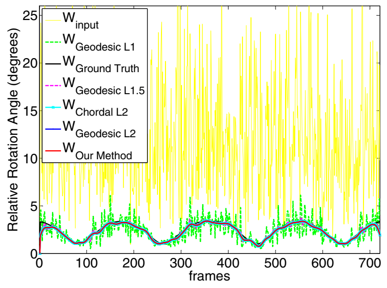
```
1: Compute $C_e = \sum_{i=1}^{n} g_i\ R_i \in \mathbb{R}^{3\times 3}$
2: Compute SVD, $C_e = U\ D\ V^T$, where diagonal
   elements of $D$ is arranged in descending order
3: if $det(UV^T) \geq 0$ then
4:     $R = UV^T$
5: else
6:     $R = U\ diag(1, 1, -1)\ V^T$
7: endif
```

IV. EXPERIMENTAL RESULT

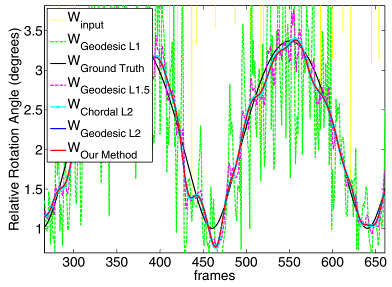
A. Simulation

The simulated ground truth data is obtained by having a combination of sinusoidal and constant change in each 'ZYX' rotation angles (simulated smooth motion), then a Gaussian noise with standard deviation of 0.1 radian is added to each of the rotation angles. This is then transformed into Rotation matrix using MATLAB's in-build function, *angle2dcm* to obtain the simulated noisy input.

To represent the smoothness of the rotation (orientation) sequence, in Fig. 4 we plotted the relative rotational angle (distance metric chosen) between consecutive orientations.



(a) Whole Sequence



(b) Zoom In

Fig. 4: Simulation Result of Relative Rotational Angle of Consecutive Orientations, Window Size = 65, Standard Deviation = 8

From Fig. 4, we can see that our method has successfully produce a smoothed orientation data (red) very close to the ground truth (black).

In order to determine how close our Pairwise method approximate Geodesic L_2 mean, we can check the norm of $\mathbf{r}$ in Line 3 of Algorithm 2. In Fig. 5, we can see that the Pairwise average method is accurate up to a tolerance, $\epsilon < 10^{-3}$, while Chordal L2 Mean method is up to 7 times less accurate.

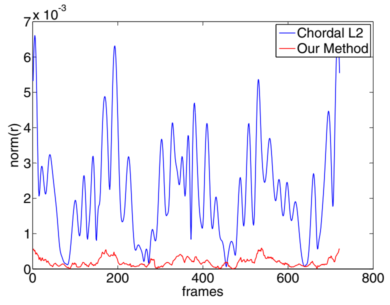
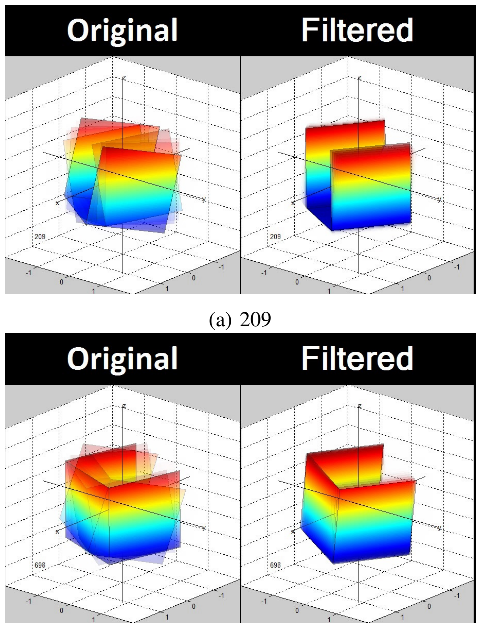


Fig. 5: Simulation Result of Difference to Geodesic L_2 Mean as Illustrated by Norm of $\boldsymbol{r}$ in Algorithm 2, Window Size = 65, Standard Deviation = 8

Fig. 6 shows the simulation result by superimposing previous frames onto the current frame to produce artificial motion blur. More motion blur corresponds to a shaky video.



(a) 209

(b) 698

Fig. 6: Simulation Result at the Corresponding Frames using Our Pairwise Average Method, where the Motion is Represented by Motion Blur

B. *Video Stabilisation - Walking Sequence*

In most mobile robotics systems, inertial measure unit (IMU) is a crucial component for estimation of the robot's relative location and orientation. The IMU is also present in most smartphones these days. In the following experiments, the gyroscope in IMU is used to estimate the camera orientation at each image frames.

By using the video sequence tested by Jia and Evans [1], we compare our result in this subsection. Similar to their method, we assume the input video sequence has undergone rolling shutter rectification.

The camera is assumed to follow a pure rotational camera model, and the difference between the smoothed and original camera orientation is used to warp the input video by a Homography (projective transformation).

We know that the Gaussian kernel's standard deviation, σ is related to the factor α in (1). Thus, we tune the σ until our smoothed curve lies close to that obtained by Jia and Evans method [1].

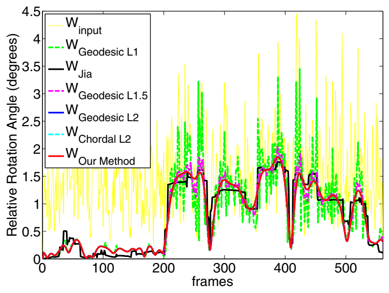


Fig. 7: Relative Rotational Angle of Consecutive Orientations in Our Pairwise Smoothing Result (Red) VS Jia and Evans's Smoothing Result (Black), and Other Window-based Smoothing Methods (Test Video in [1])

In Fig. 7, we can see that the result from our pairwise method (Red) is smoother than Jia and Evans's result (Black). This could be due to the extra constraint included by Jia and Evans to restrict the maximum angular deviation from the input rotation, as can be seen from Fig. 8.

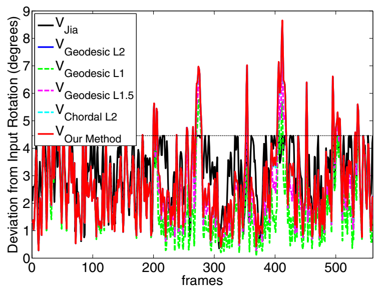


Fig. 8: Rotational Angle Deviation from Input in Our Pairwise Smoothing Result (Red), Jia and Evans's Smoothing Result(Black), and Other Window-based Smoothing Methods (Test Video in [1])

The maximum angular deviation is added to ensure that the warped frame has an upper limit on the amount of black border intruding into the view. This was done by reprojection of the gradient to be within the set bound [1]. We did not have this because we found that our method produces little, instantaneous black border intrusion (<5 continuous frames, or $<0.167s$) to justify the extra computation.

Fig. 9 contains a boxplot showing the perturbation represented by the relative angle (geodesic distance) between consecutive orientations. From this figure, we can also see that Jia and Evans method produces a result with higher median than the other window-based methods that we have proposed.

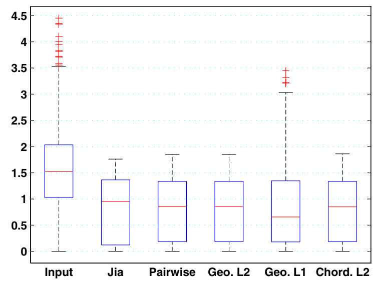


Fig. 9: Boxplot Showing the Distribution of the Relative Angle between Consecutive Orientations (Test Video in [1]). Red line is the median of the distribution, top and bottom line of the box represents 75th and 25th percentiles respectively, and red "+" shows the outliers

From the third plot in Fig. 10, we can also see that our method follows the mean of the input (blue curve) more closely than Jia and Evans's method.

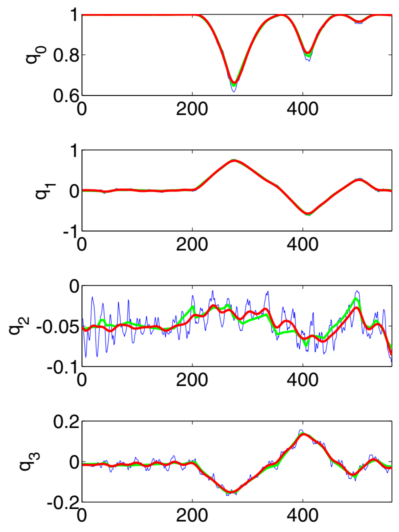


Fig. 10: Quaternion Representation of Input (Blue), Jia and Evans's Result (Green), and Our Pairwise Method (Red) (Test Video in [1])

We have also implemented the Pairwise Average method, Geodesic L_2 Mean, and Chordal L_2 Mean in C++ to compare the computational speed between the three smoothing methods. These are included in parenthesis of the last column in Table I.

The matrix operations in C++ are programmed with the help of Eigen 3.2.4 library [24]. It is noted that the C++ implementation has lower precision than MATLAB's implementation, and the iterative Geodesic L_2 Mean needs a higher tolerance, ϵ, and setting a maximum number of iterations for the program to converge.

In C++ implementation of Geodesic L_2 Mean, $\epsilon = 10^{-3}$, and maximum number of iterations is set to 6, whereas

TABLE I: Comparisons of Our Pairwise Method, Jia and Evan's, Geodesic L_2, Geodesic L_1, Geodesic $L_{1.5}$ Mean, and Chordal L_2 Mean on the video used in [1]. The Geodesic L_2 Distance is the Square of Relative Rotational Angle between Consecutive Frames. Numbers in Parenthesis is the Computation Time in C++ Implementation, Non-Parenthesised are MATLAB Implementation

	Geodesic L_2 Distance Sum		For 561 Frames
	Relative Rotation	Deviation from Input	Comp. Time (s)
Input	1737.96	0	-
Pairwise Method	**532.61**	5382.8	4.23 (**0.79**)
Jia and Evans's	559.96	6067.5	28.88
Geodesic L_2	**532.96**	5367.3	4.50 (1.44)
Geodesic L_1	657.89	3301.8	82.37
Geodesic $L_{1.5}$	554.68	4550.2	34.88
Chordal L_2	533.98	5322.9	1.21 (**0.01**)

MATLAB implementation has $\epsilon = 10^{-6}$ with no upper bound on maximum number of iterations.

Fig. 11, Fig. 12 and Fig. 13 shows the feature trajectories in the next 10 frames to visualise the difference in camera motion after stabilization.

C. Video Stabilisation - Standing Sequence

Another experiment was conducted using a Sony Xperia Z2 smartphone and its onboard gyroscope. A short video is taken by a person at a T-junction making occasional panning of the camera. This video represents a slightly different camera motion, which examines videos with a smaller magnitude camera shake than the previous video.

Different stabilization methods are tested, along with the same parameters used for the previous video. Fig. 14 shows the smoothness metric, while Fig. 15 shows the deviation from the original path.

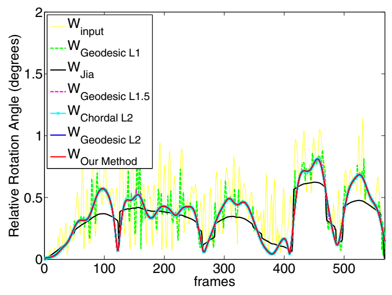
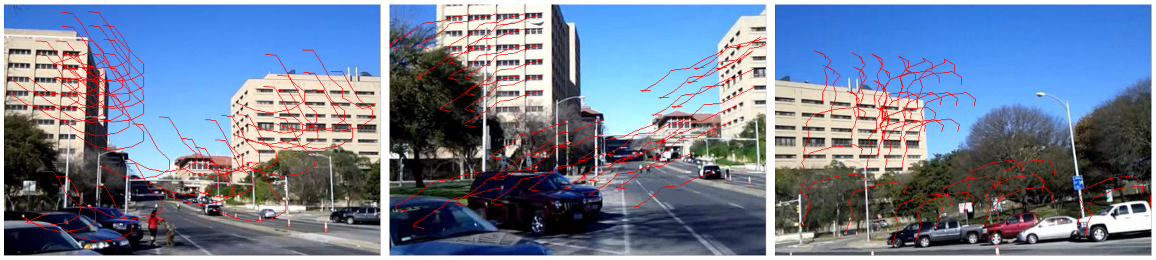


Fig. 14: Relative Rotational Angle of Consecutive Orientations in Our Pairwise Smoothing Result (Red) VS Jia and Evans's Smoothing Result (Black), and Other Window-based Smoothing Methods (Standing Sequence)

From Fig. 15, it is clear that Jia and Evan's method has overly compensated for the camera motion, since there are large and wide peaks that corresponds to motion that are not caused by noise but are removed by their method. On the other hand, the other window-based averaging method remove only the noisy camera motion.

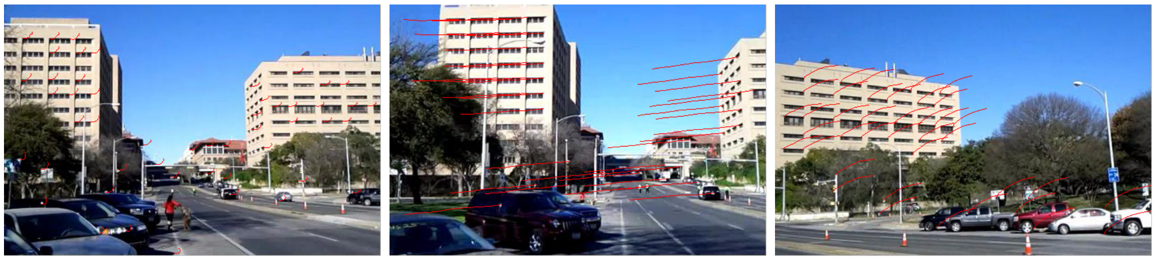
Table II shows the comparison of the two performance metrics between different rotation smoothing methods tested.

From Table II, we can again observe that although Jia and Evan's method produces a result that is smoother, it deviates from the input camera motion a lot more than the other window-based methods.



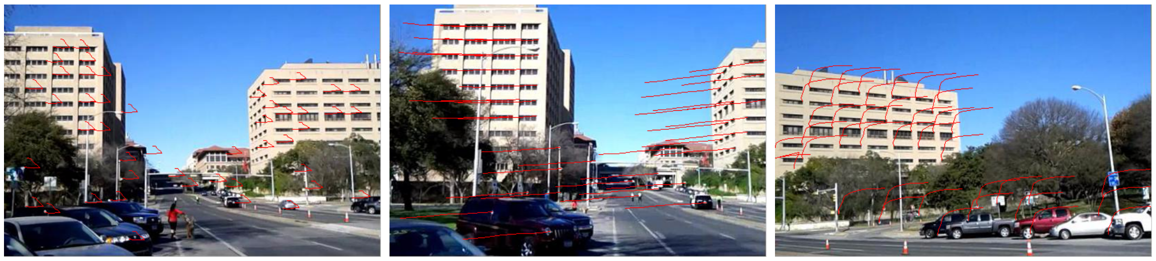
(a) 36 (b) 456 (c) 502

Fig. 11: Video Stabilisation Input Video (used in[1]) at the Corresponding Frames



(a) 36 (b) 456 (c) 502

Fig. 12: Video Stabilisation Result with Our Pairwise Method at the Corresponding Frames



(a) 36 (b) 456 (c) 502

Fig. 13: Video Stabilisation Result with Jia and Evans's Method at the Corresponding Frames

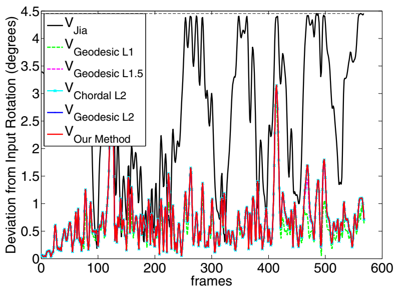


Fig. 15: Rotational Angle Deviation from Input in Our Pairwise Smoothing Result (Red), Jia and Evans's Smoothing Result(Black), and Other Window-based Smoothing Methods (Standing Sequence)

TABLE II: Comparisons of Our Pairwise Method, Jia and Evan's, Geodesic L_2, Geodesic L_1, Geodesic $L_{1.5}$ Mean, and Chordal L_2 Mean on the standing video sequence. The Geodesic L_2 Distance is the Square of Relative Rotational Angle between Consecutive Frames. Numbers in Parenthesis is the Computation Time in C++ Implementation, Non-Parenthesised are MATLAB Implementation

	Geodesic L_2 Distance Sum		For 568 Frames
	Relative Rotation	Deviation from Input	Comp. Time (s)
Input	159.87	0	-
Pairwise Method	98.43	419.17	4.87 (**0.78**)
Jia and Evans's	**72.51**	3783.30	28.96
Geodesic L_2	98.47	417.54	4.15 (1.46)
Geodesic L_1	104.01	319.36	20.87
Geodesic $L_{1.5}$	98.77	407.17	4.56
Chordal L_2	98.49	416.74	1.63 (**0.01**)

The resulting video from using Jia and Evan's method also look very similar to the one obtained using the window-based method. However, Jia and Evan's method has larger black border intrusion for this video sequence.

Frame grab of the video is not included due to space limitation, but the video will be included in the supplementary material.

V. CONCLUSION

In this paper, we have proposed a method to smooth a noisy input in the Special Euclidean Group, $\mathbf{SE(3)}$. The translational part of $\mathbf{SE(3)}$ can be smoothed by simple vector convolution with a Gaussian Kernel, and we have proposed an analogous method to smooth input rotation in the Special Orthogonal Group, $\mathbf{SO(3)}$.

We have shown that the pairwise average method (Section III-B) is superior to the method presented by Jia and Evans [1] in rotation smoothing. The pairwise averaging method presented is shown to closely approximates the weighted L_2 mean method (Section III-C), while being approximately $1.8\times$ faster in computation speed.

We have also proposed an alternative method that has a much shorter computation time, called Gaussian Weighted Chordal L_2 Mean (Section III-E). Table I and II summarises the experimental results between different rotation smoothing methods.

Additionally, it was also showed that the pairwise method successfully minimises both the relative orientation angle in consecutive frame (smoothness metric), while maintaining small deviation from the input rotation sequence. This is a trade-off controlled by the Gaussian kernel's standard deviation, σ, similar to the scalar factor, α in (1) presented in [1].

Due to the use of Gaussian Convolution, the method does not introduce drift and scale changes (sum of weights = 1), and is temporally invariant. The pairwise computation is also parallelisable, fast ($\approx 1.4ms$/data point), and have deterministic computation time.

Unlike iterative method, it also does not require any initial estimate, which may affect the convergence of the iterative algorithm.

The method allows smoothing of a noisy motion in real-time. One drawback of the proposed Pairwise Rotations Averaging method is the delay introduced by the technique, which is half the window length used in the convolution.

However, with the extra information about the future motion, we can ensure that the solution will stay close to the mean of the actual trajectory, as shown in the quaternion plot (Fig. 10)

Finally, the video stabilisation output of the walking sequence is illustrated in features trajectory shown in Fig. 12. Videos of the stabilisation results and comparisons will also be attached as supplementary material.

Future extension to this work include the use orientation (rotations) input from other sources (e.g. visual odometry), and compare the results with orientation obtained through the IMU, and results using other video stabilisation methods.

Instead of using homography for the assumption of purely rotating camera, we can also use spatially-varying warp to synthesise novel views for the purpose of stabilizing a video with large translational vibration, similar to those discussed in [25] and [26].

In addition, since there are a lot of parallelisable operations, the proposed method may also be implemented on a GPU for faster computation.

References

[1] C. Jia and B. Evans, "Constrained 3d rotation smoothing via global manifold regression for video stabilization," *Signal Processing, IEEE Transactions on*, vol. 62, no. 13, pp. 3293–3304, July 2014.

[2] X. Hou and R. Mahony, "Dynamic kinesthetic boundary for haptic teleoperation of aerial robotic vehicles," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 4549–4950.

[3] R. Laird, M. Bruch, M. West, D. Ciccimaro, and H. Everett, "Issues in vehicle teleoperation for tunnel and sewer reconnaissance," DTIC Document, Tech. Rep., 2000.

[4] P. Ballou, "Improving pilot dexterity with a telepresent rov," in *Proceedings of the Vehicle Teleoperation Interfaces Workshop, IEEE ICRA*, 2001.

[5] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 15–22.

[6] R. Borja, J. de la Pinta, A. lvarez, and J. Maestre, "Integration of service robots in the smart home by means of upnp: A surveillance robot case study," *Robotics and Autonomous Systems*, vol. 61, no. 2, pp. 153 – 160, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889012001881

[7] Q. Wang and R. K. Ward, "Fast image/video contrast enhancement based on weighted thresholded histogram equalization," *Consumer Electronics, IEEE Transactions on*, vol. 53, no. 2, pp. 757–764, 2007.

[8] J. M. Boyce, A. M. Tourapis, and J. A. Cooper, "Encoding method and apparatus enabling fast channel change of compressed video," Aug. 19 2014, uS Patent 8,811,492.

[9] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, "Full-frame video stabilization with motion inpainting," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 7, pp. 1150–1163, 2006.

[10] S. Ertürk, "Real-time digital image stabilization using kalman filters," *Real-Time Imaging*, vol. 8, no. 4, pp. 317–328, 2002.

[11] R. Hartley, K. Aftab, and J. Trumpf, "L1 rotation averaging using the weiszfeld algorithm," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, vol. 0, pp. 3041–3048, 2011.

[12] C. Jia and B. Evans, "3d rotational video stabilization using manifold optimization," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, May 2013, pp. 2493–2497.

[13] J. Fredriksson and C. Olsson, "Simultaneous multiple rotation averaging using lagrangian duality," in *Computer Vision ACCV 2012*, ser. Lecture Notes in Computer Science, K. Lee, Y. Matsushita, J. Rehg, and Z. Hu, Eds. Springer Berlin Heidelberg, 2013, vol. 7726, pp. 245–258.

[14] K. Aftab, R. Hartley, and J. Trumpf, "Generalized weiszfeld algorithms for lq optimization," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 37, no. 4, pp. 728–745, April 2015.

[15] A. Chatterjee and V. Govindu, "Efficient and robust large-scale rotation averaging," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, Dec 2013, pp. 521–528.

[16] J. Glover and L. P. Kaelbling, "Tracking 3-d rotations with the quaternion bingham filter," 2013.

[17] M. Moakher, "Means and averaging in the group of rotations," *SIAM Journal on Matrix Analysis and Applications*, vol. 24, no. 1, pp. 1–16, 2002. [Online]. Available: http://dx.doi.org/10.1137/S0895479801383877

[18] K. Shoemake, "Animating rotation with quaternion curves," *In Computer Graphics: Proceedings of SIGGRAPH 85*, pp. 245–254, 1985.

[19] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," 2012.

[20] A. Sarlette and R. Sepulchre, "Consensus optimization on manifolds," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 56–76, 2009.

[21] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, Mar 1966. [Online]. Available: http://dx.doi.org/10.1007/BF02289451

[22] R. Everson, "Orthogonal, but not orthonormal, procrustes problems," in *Advances in Computational Mathematics . (Submitted). Available from http://www.ee.ic.ac.uk/research/neural/everson*, 1997.

[23] G. Kurz, I. Gilitschenski, S. Julier, and U. D. Hanebeck, "Recursive estimation of orientation based on the bingham distribution," *CoRR*, vol. abs/1304.8019, 2013. [Online]. Available: http://arxiv.org/abs/1304.8019

[24] B. Jacob and G. Guennebaud, "Eigen @ONLINE," 2015, accessed: 2015-04-21.

[25] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3d video stabilization," in *ACM SIGGRAPH 2009 Papers*, ser. SIGGRAPH '09. New York, NY, USA: ACM, 2009, pp. 44:1–44:9. [Online]. Available: http://doi.acm.org/10.1145/1576246.1531350

[26] J. Kopf, M. F. Cohen, and R. Szeliski, "First-person hyper-lapse videos," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 78:1–78:10, July 2014. [Online]. Available: http://doi.acm.org/10.1145/2601097.2601195