Robust Relative Rotation Averaging

Avishek Chatterjee and Venu Madhav Govindu

Abstract—This paper addresses the problem of robust and efficient relative rotation averaging in the context of large-scale Structure from Motion. Relative rotation averaging finds global or absolute rotations for a set of cameras from a set of observed relative rotations between pairs of cameras. We propose a generalized framework of relative rotation averaging that can use different robust loss functions and jointly optimizes for all the unknown camera rotations. Our method uses a quasi-Newton optimization which results in an efficient iteratively reweighted least squares (IRLS) formulation that works in the Lie algebra of the 3D rotation group. We demonstrate the performance of our approach on a number of large-scale data sets. We show that our method outperforms existing methods in the literature both in terms of speed and accuracy.

Index Terms—Relative Rotation Averaging, Structure from Motion, 3D Rotation Group, SO(3), Iteratively Reweighted Least Squares, Quasi-Newton Optimization, Gauss-Newton Optimization.

1 Introduction

Solving the structure-from-motion (henceforth SfM) problem involves recovering both the 3D structure of points as well as the camera motions from corresponding or matched points across multiple images. The most popular approach is a non-linear optimization method known as bundle-adjustment (henceforth BA) that fits a generative model for both 3D structure and cameras to the observed matched image points [1]. BA works by minimizing reprojection errors on the camera plane and is statistically optimal. Being a non-linear optimization, BA requires a good initialization since it is only guaranteed to converge to a local minimum. This requirement of a good initialization becomes increasingly stringent with growing problem size. Additionally, since it involves a highly complex, non-linear optimization over many unknowns, BA is plagued by the problems of robustness to outliers and failure to converge owing to the complexity of the energy landscape. Using efficient optimizers and a variety of heuristics, modern SfM pipelines [2], [3], [4], [5] are able to solve the BA problem for a large number of images. To mitigate the problem of robustness, many approaches incrementally grow the SfM solution one-camera-at-a-time instead of using a batch optimization of all cameras at once [2], [4]. This makes the problem quite slow and BA solutions are not easily scalable to large-scale problems as they involve a very high computational cost and expensive hardware.

While BA seeks to simultaneously solve for structure and motion using many images, it may be observed that methods based on epipolar geometry can recover camera geometry independent of the 3D structure which is factored out of the problem. We can extend this property of structure-independent geometry estimation to an arbitrary number of cameras using the approach of motion averaging [6]. Apart from reducing the number of unknowns involved, by defining a problem purely in terms of the cameras, motion averaging utilizes the rich geometric structure of finite-dimensional Lie groups [7]. Using the two-frame epipolar geometry we can only recover the heading direction of the second camera with respect to the first and not the scale of the actual 3D translation between them. However, we can fully recover the 3D relative rotation between a camera pair using their epipolar geometry. Consequently, a common approach has been to first solve the motion averaging problem for 3D rotations alone and utilize this solution in the estimation of 3D translation [8], [9]. Once the motions of the cameras are solved, it is significantly easier to find the positions of the 3D structure points using multiview triangulation [10] which provides a good initial guess for BA solutions [9], [11].

In this paper, we address the problem of averaging relative 3D rotations between camera pairs in an SfM context. Apart from solving large-scale problems, the desiderata for our solution are : efficiency, accuracy and robustness. Our earlier solution in [12], achieved these requirements and was derived from the original approach of [6]. While [6] was non-robust as it used the ℓ_2 metric in its computations, [12] achieved robustness and accuracy by utilizing an ℓ_1-based optimization followed by an iteratively reweighted least-squares (IRLS) refinement in the Lie-algebra of the group of 3D rotations. In this paper, we generalize and improve upon the solution in [12] in several significant ways :

1) We show that the relative rotation averaging solutions in [6], [12] are quasi-Newton optimizations.
2) While in [12], the IRLS step was based on the Geman-McClure function, in this paper we show how to utilize and evaluate the performance of different robust loss functions. We provide a fairly exhaustive evaluation of the performance of a variety of robust loss functions that are popular in the statistical literature. Our evaluation leads to a recommendation of the $\ell_{\frac{1}{2}}$ loss function that provides the best empirical performance.
3) We demonstrate the efficacy of our method in terms of both accuracy and efficiency on a number of large-scale, real-world datasets and also compare its relative performance with respect to two other rotation averaging methods, i.e. DISCO [11] and the Weiszfeld method [13].

In Section 2, we briefly discuss properties of 3D rotations of relevance to this paper. The problem of relative rotation averaging is defined in Section 3. Our technique of averaging relative rotations is introduced next in Section 4. In Section 4 we also present a generalized framework of joint averaging of relative rotations under various robust metrics on 3D rotations. Subsequently, in Section 5 we briefly survey the other existing methods for relative rotation averaging and also discuss some scenarios under which the distributed averaging approach of [13] fails. In Section 6, we demonstrate the accuracy and efficacy of our method on a number of large-scale real world datasets and we present some conclusions in Section 7.

2 Properties of 3D Rotations

In this section, we briefly recapitulate some of the properties of 3D rotations that play a role in our formulation. Of the many representations of 3D rotations available in the literature, we use the 3×3 orthonormal matrix $\mathbf{R}$, i.e. $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}_3$ where $\mathbf{I}_3$ is the 3×3 identity matrix. Moreover, it is required that $det(\mathbf{R}) = +1$. The set of all such 3×3 orthonormal matrices with determinant equal to $+1$ forms a closed group, i.e. the Special Orthogonal group of dimension 3 or $SO(3)$. The reader may refer to [14] for details. In addition to being a group, $SO(3)$ is equipped with the smooth differentiable structure of a Riemannian manifold, i.e. $SO(3)$ is a finite-dimensional Lie group. Both the group operations of product and inverse are smooth differentiable mappings for such Lie groups [6]. For our purposes, we recognise that the local neighborhood of a point on the group is adequately described by its tangent space known as the Lie algebra. Additionally, there exist direct mappings between this Lie algebra and the corresponding Lie group and vice-versa, which are specified by the exponential and logarithm functions. For $SO(3)$, the corresponding Lie algebra is denoted as $\mathfrak{so}(3)$.

Choosing the canonical or Euclidean metric in the Lie algebra representation leads to an intuitive definition of the geodesic distance on the rotation group $SO(3)$. This distance is easily specified by the familiar axis-angle parametrization of 3D rotations. In the axis-angle form, the 3×1 vector $\boldsymbol{\omega}$ represents a rotation of angle $\|\boldsymbol{\omega}\|$ about the axis $\boldsymbol{\omega}/\|\boldsymbol{\omega}\|$. The relation between the matrix representation of rotation $\mathbf{R}$ and its axis-angle representation $\boldsymbol{\omega}$ is given by

$$\mathbf{R} = \exp([\boldsymbol{\omega}]_\times) \quad (1)$$

$$[\boldsymbol{\omega}]_\times = \log(\mathbf{R}) \quad (2)$$

where $[\boldsymbol{\omega}]_\times$ is the skew-symmetric form of $\boldsymbol{\omega}$, i.e.

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \quad (3)$$

It will be immediately noted that the skew-symmetric form $[\boldsymbol{\omega}]_\times$ defines the Lie algebra for the rotation group $SO(3)$. For notational simplicity, throughout this paper we use $\mathbf{R}(\boldsymbol{\omega})$ to denote the 3D rotation obtained by the exponential mapping applied to $[\boldsymbol{\omega}]_\times \in \mathfrak{so}(3)$ and $\boldsymbol{\omega}(\mathbf{R})$ to denote the 3×1 vector representation of $\boldsymbol{\omega}$ extracted from the logarithm applied to $\mathbf{R} \in SO(3)$. We also note that for the rotation group $SO(3)$, the exponential and logarithm functions have closed forms known as the Rodrigues formula [14].

Another property of relevance to us arises out of the non-commutative nature of the $SO(3)$ group. Specifically, for two 3D rotations about different axes, $\mathbf{R}(\boldsymbol{\omega}_1)\mathbf{R}(\boldsymbol{\omega}_2) \neq \mathbf{R}(\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2)$. Instead, $\mathbf{R}(\boldsymbol{\omega}_1)\mathbf{R}(\boldsymbol{\omega}_2) = \mathbf{R}(BCH(\boldsymbol{\omega}_1, \boldsymbol{\omega}_2))$ where $BCH(.,.)$ is the Baker-Campbell-Hausdorff form expressed as a series on the two Lie algebra terms $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ [6], [15]. For the rotation group $SO(3)$, the BCH series has a closed form expression $BCH(\boldsymbol{\omega_1}, \boldsymbol{\omega_2}) = \alpha\boldsymbol{\omega_1} + \beta\boldsymbol{\omega_2} + \gamma\,\boldsymbol{\omega_1} \times \boldsymbol{\omega_2}$, where the scalars α, β and γ are functions of $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ [16], [17].

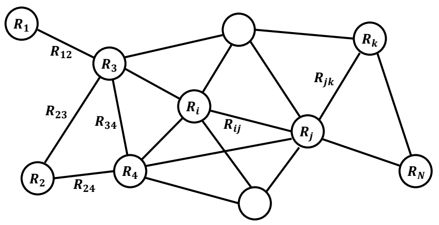


Fig. 1. A view-graph representation of the available relative rotations. The averaging problem considered in this paper is one of recovering the rotations at the vertices (cameras) given the relative rotations on the edges.

Given the Lie group structure of $SO(3)$, the intrinsic or Riemannian distance between two rotations $\mathbf{R}_1$ and $\mathbf{R}_2$ can be easily specified. In terms of the axis-angle parametrization, this distance is the magnitude of the angle that takes us from $\mathbf{R}_1$ to $\mathbf{R}_2$. Noting that the distance metric on $SO(3)$ is bi-invariant in nature [14], we have

$$\begin{aligned} & d(\mathbf{R}_1, \mathbf{R}_2) = d(\mathbf{R}_2, \mathbf{R}_1) = d(\mathbf{I}, \mathbf{R}_2\mathbf{R}_1^{-1}) \\ = \; & \frac{1}{\sqrt{2}}\|\log(\mathbf{R}_2\mathbf{R}_1^{-1})\|_F = \|\boldsymbol{\omega}(\mathbf{R}_2\mathbf{R}_1^{-1})\| \\ = \; & \|BCH(\boldsymbol{\omega}_2, -\boldsymbol{\omega}_1)\| \end{aligned} \quad (4)$$

where $d(\cdot, \cdot)$ is the Riemannian distance on $SO(3)$ and $\|\cdot\|_F$ denotes the Frobenius norm.

3 Relative Rotation Averaging

With these preliminaries, we can now define the problem of robust relative rotation averaging on the $SO(3)$ group. Consider Figure 1 which depicts a graph (known as the view-graph) $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ representing the geometric relationships between cameras viewing a scene. Each vertex in $\mathcal{V}$ represents a camera with an unknown absolute rotation. We denote the number of vertices or cameras i.e. $|\mathcal{V}|$ as N and the number of edges i.e. $|\mathcal{E}|$ as M. The presence of an edge $(i, j) \in \mathcal{E}$ implies that the relative rotation between cameras i and j is available. The set of all 3D rotations $\mathbf{R}_\mathcal{V} = \{\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_N\}$ completely specifies the *global* rotations of all the cameras with respect to a given frame of reference. Additionally, we may denote the *relative* rotation between two cameras i and j as $\mathbf{R}_{ij}$ and denote the collection of all relative rotations, one for each edge, as $\mathbf{R}_\mathcal{E}$. If the set of edges $\mathcal{E}$ span the entire view-graph, then we can solve for the unknown absolute rotations of all the cameras from the pairwise relative rotations. In other words, while in principle M can be as large as ${}^NC_2 = \frac{N(N-1)}{2}$, we only need $M = (N-1)$ suitable edges to solve for $\mathbf{R}_\mathcal{V}$. We note that if images i and j share a sufficient number of common feature points, then using the epipolar geometry [18], [19], [20] we can obtain an estimate of $\mathbf{R}_{ij}$.

Since the cameras i and j have absolute rotations of $\mathbf{R}_i$ and $\mathbf{R}_j$ respectively in a given frame of reference, it is obvious that the relative rotation between them should obey the relationship

$$\mathbf{R}_{ij} = \mathbf{R}_j\mathbf{R}_i^{-1}, \ \forall\{i,j\} \in \mathcal{E} \tag{5}$$

The problem of relative rotation averaging can be stated as follows: Given a sufficient number of relative rotations $\mathbf{R}_{ij} \in \mathbf{R}_{\mathcal{E}}$ (lhs of Equation 5) we seek an estimate of the global camera rotations, i.e. $\mathbf{R}_{\mathcal{V}}$. In practice, we always have a larger number of edges than is required to span the view-graph, i.e. $|\mathcal{E}| > N-1$, implying that we have a redundant set of observations. We also note that due to the presence of noise or outliers the set of relative rotations are inconsistent, i.e. we cannot find a solution $\mathbf{R}_{\mathcal{V}} = \{\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_N\}$ that exactly satisfies all the constraints $\{\mathbf{R}_{ij} = \mathbf{R}_j\mathbf{R}_i^{-1} | \forall(i,j) \in \mathcal{E}\}$. Therefore, we seek to find an estimate $\mathbf{R}_{\mathcal{V}}$ that is most consistent with the observed relative rotations. This solution is obtained by minimizing a cost function that penalizes the discrepancy between the observed relative rotations $\mathbf{R}_{ij}$ and that suggested by the estimate $\mathbf{R}_j\mathbf{R}_i^{-1}$, i.e.

$$\mathbf{R}_{\mathcal{V}} = \underset{\{\mathbf{R}_1,\mathbf{R}_2,\cdots,\mathbf{R}_N\}}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{E}} \rho(d(\mathbf{R}_{ij}, \mathbf{R}_j\mathbf{R}_i^{-1})) \tag{6}$$

where $d(\cdot)$ is a distance measure between two rotations in $SO(3)$ and $\rho(\cdot)$ is a loss function defined over this distance measure.

3.1 Choice of Loss Function $\rho(.)$

The distance metric $d(\cdot,\cdot)$ described above in Section 2 can be used to measure the residual fitting error for a relative rotation observation on a single edge in $\mathcal{E}$. However, to define the objective function to be minimized in Equation 6, we also need to define a loss function $\rho(\cdot)$ that determines the relative contributions of individual errors. The original approach of [6] uses a loss function of $\rho(x) = x^2$ in Equation 6 resulting in a non-robust method as it is well-known that the quadratic loss function is highly susceptible to outliers. Contemporary SfM pipelines are designed to solve large-scale problems using many real-world images. It is often the case that the existence of repeated elements in many man-made structures such as building facades, variation in illumination etc., all result in false feature point matches across image pairs. As a result, even when one uses robust techniques for epipolar geometry estimation, one does end up with corrupted relative rotation estimates $\mathbf{R}_{ij}$ for some of the edges in the viewgraph.

In statistical terms, the quadratic loss function $\rho(x) = x^2$ has an unbounded influence function [21], as a result any observation that is far from the estimate will result in a very high penalty. Although, unlike Euclidean space, the distance on the manifold of $SO(3)$ is bounded by π radians, this bound is always very large compared to the noise level in estimated relative rotations. Consequently, the optimization will significantly move the estimate away from the true solution, i.e. the resulting estimate will be severely corrupted by any outliers. To some extent this problem is mitigated by using an ℓ_1 loss function, i.e. $\rho(x) = |x|$. This is the approach taken by the Weiszfeld method of [13]. However, while the ℓ_1 penalty makes it robust, the distributed averaging approach of [13] suffers from the problem of slow convergence for large-scale datasets.

The ℓ_1 function has an influence function of constant magnitude, hence it is more robust than the ℓ_2 loss function. However, in an ideal scenario, we would like to allow no influence for the measurements which are outliers. While this is impossible to achieve without classifying observations as outliers, we have recourse to loss functions $\rho(x)$ other than ℓ_1 that have diminishing influence as $|x|$ increases. In Table 1, we provide a fairly exhaustive list of loss functions that are common in the robust statistics literature [21]. While loss functions such as Geman-McClure have a diminishing influence that provides robustness, such methods are more susceptible to converge to an improper local minima. As a result, the optimization procedure needs to be initialized with a good initial guess. Often in the literature of robust statistics [22], such initializations are obtained using the solution of another loss function e.g. the ℓ_1 loss function.

Before we move on to describing our approach for robust averaging of relative rotations, we wish to remark on the choice of distance functions on the space of 3D rotations. While we have used the geodesic distance on $SO(3)$ as the distance metric in this paper, there are a few other distances used in the literature. Some approaches ignore the intrinsic constraints that define a rotation as an element of the $SO(3)$ group and compute the extrinsic distance instead. In other words, these methods use the distance $d_{chordal}(\mathbf{R}_1, \mathbf{R}_2) = \|\mathbf{R}_1 - \mathbf{R}_2\|_F$ which is also known as the chordal distance. Rotations can also be represented as unit quaternions [14]. If the rotations $\mathbf{R}_1$ and $\mathbf{R}_2$ are equivalently represented by the unit quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$, then the quaternion distance between them is defined as $d_{quaternion}(\mathbf{R}_1, \mathbf{R}_2) = \min(\|\mathbf{q}_1 + \mathbf{q}_2\|), (\|\mathbf{q}_1 - \mathbf{q}_2\|)$. If we denote the geodesic distance between two rotations $\mathbf{R}_1$ and $\mathbf{R}_2$ as θ, then the chordal distance between them $d_{chordal}(\mathbf{R}_1, \mathbf{R}_2) = \|\mathbf{R}_1 - \mathbf{R}_2\|_F$ is related to the geodesic distance as $d_{chordal}(\mathbf{R}_1, \mathbf{R}_2) = 2\sqrt{2}\sin(\theta/2)$. Equivalently, the quaternion distance is related to the geodesic distance as $d_{quaternion}(\mathbf{R}_1, \mathbf{R}_2) = 2\sin(\theta/4)$. All of these distances can also be subsumed under our approach.

4 Our Approach

In this section, we describe our approach to solving the optimization problem in Equation 6. In brief, we reduce the problem in Equation 6 into an iterative optimization problem. In each iteration, this optimization problem is solved using a quasi-Newton method that leads to an overall averaging method that is both efficient and robust to outliers.

4.1 Reduction to Iteratively Reweighted Least Squares

The optimization problem in Equation 6 can be rewritten as

$$\begin{aligned}
\mathbf{R}_{\mathcal{V}} &= \underset{\{\mathbf{R}_1,\mathbf{R}_2,\cdots,\mathbf{R}_N\}}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{E}} \rho(d(\mathbf{R}_{ij}, \mathbf{R}_j\mathbf{R}_i^{-1})) \\
&= \underset{\{\mathbf{R}_1,\mathbf{R}_2,\cdots,\mathbf{R}_N\}}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{E}} \rho(\|\boldsymbol{\omega}(\mathbf{R}_j^{-1}\mathbf{R}_{ij}\mathbf{R}_i)\|) \\
&= \underset{\{\mathbf{R}_1,\mathbf{R}_2,\cdots,\mathbf{R}_N\}}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{E}} \rho(\|\boldsymbol{\omega}(\Delta\mathbf{R}_{ij})\|)
\end{aligned} \tag{7}$$

Name	Loss function ($\rho(x)$)	Influence function ($\psi(x)$)	Weight function ($\phi(x)$)
ℓ_2	$x^2/2$	x	1
ℓ_1	$\lvert x\rvert$	$sign(x)$	$1/\lvert x\rvert$
ℓ_α	$\lvert x\rvert^\alpha/\alpha$	$sign(x)\lvert x\rvert^{\alpha-1}$	$\lvert x\rvert^{\alpha-2}$
Geman-McClure	$\frac{x^2/2}{\alpha^2+x^2}$	$\frac{\alpha^2 x}{(\alpha^2+x^2)^2}$	$\frac{\alpha^2}{(\alpha^2+x^2)^2}$
Huber	$\begin{cases} x^2/2 & \text{if } \lvert x\rvert \le \alpha \\ \alpha(\lvert x\rvert-\alpha/2) & \text{if } \lvert x\rvert \ge \alpha \end{cases}$	$\begin{cases} x & \text{if } \lvert x\rvert \le \alpha \\ \alpha sign(x) & \text{if } \lvert x\rvert \ge \alpha \end{cases}$	$\begin{cases} 1 & \text{if } \lvert x\rvert \le \alpha \\ \alpha/\lvert x\rvert & \text{if } \lvert x\rvert \ge \alpha \end{cases}$
Pseudo-Huber or $\ell_1-\ell_2$	$\alpha^2\left(\sqrt{1+x^2/\alpha^2}-1\right)$	$\frac{x}{\sqrt{1+x^2/\alpha^2}}$	$\frac{1}{\sqrt{1+x^2/\alpha^2}}$
Andrews	$\begin{cases} -\alpha^2-\alpha^2\cos(x/\alpha) & \text{if } \lvert x\rvert \le \alpha\pi \\ 0 & \text{if } \lvert x\rvert \ge \alpha\pi \end{cases}$	$\begin{cases} \alpha\sin(x/\alpha) & \text{if } \lvert x\rvert \le \alpha\pi \\ 0 & \text{if } \lvert x\rvert \ge \alpha\pi \end{cases}$	$\begin{cases} \frac{\sin(x/\alpha)}{x/\alpha} & \text{if } \lvert x\rvert \le \alpha\pi \\ 0 & \text{if } \lvert x\rvert \ge \alpha\pi \end{cases}$
Tukey's Biweight or Bisquare	$\begin{cases} \frac{\alpha^2}{6}\left(1-\left(1-(x/\alpha)^2\right)^3\right) & \text{if } \lvert x\rvert \le \alpha \\ 0 & \text{if } \lvert x\rvert \ge \alpha \end{cases}$	$\begin{cases} x\left(1-(x/\alpha)^2\right)^2 & \text{if } \lvert x\rvert \le \alpha \\ 0 & \text{if } \lvert x\rvert \ge \alpha \end{cases}$	$\begin{cases} \left(1-(x/\alpha)^2\right)^2 & \text{if } \lvert x\rvert \le \alpha \\ 0 & \text{if } \lvert x\rvert \ge \alpha \end{cases}$
Cauchy	$\frac{\alpha^2}{2}\log\left(1+(x/\alpha)^2\right)$	$\frac{x}{1+(x/\alpha)^2}$	$\frac{1}{1+(x/\alpha)^2}$
Fair	$\alpha^2\left(\lvert x\rvert/\alpha-\log(1+\lvert x\rvert/\alpha)\right)$	$\frac{x}{1+\lvert x\rvert/\alpha}$	$\frac{1}{1+\lvert x\rvert/\alpha}$
Logistic	$\alpha^2\log(\cosh(x/\alpha))$	$\alpha\tanh(x/\alpha)$	$\frac{\tanh(x/\alpha)}{x/\alpha}$
Talwar	$\begin{cases} x^2 & \text{if } \lvert x\rvert \le \alpha \\ \alpha^2 & \text{if } \lvert x\rvert \ge \alpha \end{cases}$	$\begin{cases} x & \text{if } \lvert x\rvert \le \alpha \\ 0 & \text{if } \lvert x\rvert \ge \alpha \end{cases}$	$\begin{cases} 1 & \text{if } \lvert x\rvert \le \alpha \\ 0 & \text{if } \lvert x\rvert \ge \alpha \end{cases}$
Welsch	$\frac{\alpha^2}{2}\left(1-\exp\left(-(x/\alpha)^2\right)\right)$	$x\exp\left(-(x/\alpha)^2\right)$	$\exp\left(-(x/\alpha)^2\right)$

TABLE 1
Different robust loss functions.

where $\Delta\mathbf{R}_{ij} = \mathbf{R}_j^{-1}\mathbf{R}_{ij}\mathbf{R}_i$. It will be noted that given a current estimate of the rotations $\mathbf{R}_\mathcal{V}$, for the cameras (vertices in $\mathcal{V}$), $\|\Delta\mathbf{R}_{ij}\|$ denotes the discrepancy or difference between the observed relative rotation between cameras i and j of $\mathbf{R}_{ij}$ and that implied by the current estimate $\mathbf{R}_j\mathbf{R}_i^{-1}$. In other words, Equation 7 implies that our goal is to find the camera rotations that best explain the observed relative rotations, i.e. are most consistent with the observed data according to the chosen distance and loss functions $d(\cdot,\cdot)$ and $\rho(\cdot)$ respectively.

In the following we will consider an iterative approach to minimizing our objective function. To keep the notation simple, we drop the iteration index in our representation. Let the current estimate of $\mathbf{R}_\mathcal{V}$ be $\{\mathbf{R}_1,\cdots,\mathbf{R}_N\}$. In the current iteration, we seek to apply an update to $\mathbf{R}_\mathcal{V}$ that will decrease the cost function of Equation 7. Let this update be $\{\Delta\mathbf{R}_1,\cdots,\Delta\mathbf{R}_N\}$, i.e. after the update the new estimate of $\mathbf{R}_\mathcal{V}$ will be $\{\mathbf{R}_1\Delta\mathbf{R}_1,\cdots,\mathbf{R}_N\Delta\mathbf{R}_N\}$. Therefore, in a given iteration, our objective is to minimize

$$\sum_{(i,j)\in\mathcal{E}} \rho\left(\left\|\boldsymbol{\omega}\left(\Delta\mathbf{R}_j^{-1}\Delta\mathbf{R}_{ij}\Delta\mathbf{R}_i\right)\right\|\right) \tag{8}$$

For each individual rotation update $\Delta\mathbf{R}_i$, let the equivalent axis-angle representation be denoted as $\Delta\boldsymbol{\omega}_i$ for all $i \in \{1,\cdots,N\}$. We collect all of these vectors into a single $3N\times 1$ vector $\Delta\boldsymbol{\Omega}_\mathcal{V} = \left[\Delta\boldsymbol{\omega}_1^T,\cdots,\Delta\boldsymbol{\omega}_N^T\right]^T$. Further, for each of the residual rotation terms $\Delta\mathbf{R}_{ij}$, the equivalent axis-angle representation is denoted as $\Delta\boldsymbol{\omega}_{ij}$ for all $(i,j)\in\mathcal{E}$. The concatenation of all these vectors into a $3M\times 1$ vector is denoted as $\Delta\boldsymbol{\Omega}_\mathcal{E}$. Recalling that an orthonormal rotation matrix $\mathbf{R}$ is equal to the matrix exponentiation on the axis-angle form (i.e. Lie algebraic representation), Equation 8 boils down to finding $\Delta\boldsymbol{\Omega}_\mathcal{V}$ that minimizes the cost function $F(\Delta\boldsymbol{\Omega}_\mathcal{V})$ given by

$$F(\Delta\boldsymbol{\Omega}_\mathcal{V}) = \sum_{(i,j)\in\mathcal{E}} \rho\left(\left\|\boldsymbol{\omega}\left(\mathbf{R}\left(-\Delta\boldsymbol{\omega}_j\right)\mathbf{R}\left(\Delta\boldsymbol{\omega}_{ij}\right)\mathbf{R}\left(\Delta\boldsymbol{\omega}_i\right)\right)\right\|\right)$$
$$= \sum_{(i,j)\in\mathcal{E}} \rho\left(\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|\right) \tag{9}$$

where,

$$\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right) = \boldsymbol{\omega}\left(\mathbf{R}\left(-\Delta\boldsymbol{\omega}_j\right)\mathbf{R}\left(\Delta\boldsymbol{\omega}_{ij}\right)\mathbf{R}\left(\Delta\boldsymbol{\omega}_i\right)\right) \tag{10}$$

The gradient of $F(\Delta\boldsymbol{\Omega}_\mathcal{V})$ is given by

$$\begin{aligned} \boldsymbol{\nabla}F\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right) &= \sum_{(i,j)\in\mathcal{E}} \boldsymbol{\nabla}\rho\left(\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|\right) \\ &= \sum_{(i,j)\in\mathcal{E}} \psi\left(\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|\right)\boldsymbol{\nabla}\left(\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|\right) \end{aligned} \tag{11}$$

where $\psi(r) = \frac{\partial\rho(r)}{\partial r}$ is the influence function. Let us define $\phi\left(\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|\right) = \frac{\psi\left(\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|\right)}{\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|}$. Therefore

$$\begin{aligned} &\boldsymbol{\nabla}F\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right) \\ &= \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} \phi\left(\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|\right)2\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|\boldsymbol{\nabla}\left(\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|\right) \\ &= \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} \phi\left(\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|\right)\boldsymbol{\nabla}\left(\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|^2\right) \end{aligned} \tag{12}$$

We can find the optimal $\Delta\boldsymbol{\Omega}_\mathcal{V}$ by equating $\boldsymbol{\nabla}F\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)$ to zero. Here we use the IRLS [23] optimization and therefore, during each iteration, we treat $\phi\left(\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|\right)$ as a constant given by $\phi\left(\left\|\mathbf{r}_{ij}\left(\mathbf{0}\right)\right\|\right)$. For simplicity of notation, let us denote $\phi\left(\left\|\mathbf{r}_{ij}\left(\mathbf{0}\right)\right\|\right)$ as ϕ_{ij}. Therefore, our problem is to solve

$$\sum_{(i,j)\in\mathcal{E}} \phi_{ij}.\boldsymbol{\nabla}\left(\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|^2\right) = \mathbf{0} \tag{13}$$

which is equivalent to the following minimization problem

$$\underset{\Delta\boldsymbol{\Omega}_\mathcal{V}}{\text{minimize}} \sum_{(i,j)\in\mathcal{E}} \phi_{ij}\left\|\mathbf{r}_{ij}\left(\Delta\boldsymbol{\Omega}_\mathcal{V}\right)\right\|^2 \tag{14}$$

We reiterate that the weights $\phi_{ij} = \phi\left(\left\|\mathbf{r}_{ij}\left(\mathbf{0}\right)\right\|\right)$ are computed for each edge $(i,j)\in\mathcal{E}$ at every iteration. Thus, we have reduced our problem of relative rotation averaging into an iteratively reweighted nonlinear least squares problem. We shall now see how in each iteration, we can solve the weighted nonlinear least squares problem using a quasi-Newton method.

4.2 Optimization using Quasi-Newton Method

In the quasi-Newton method of optimization, the Hessian of the cost function is approximated by a positive semi-definite matrix. While positive semi-definiteness of the Hessian ensures that the updates are always in a descent direction, there are many possible choices of the Hessian. Note that the Gauss-Newton algorithm is a specific case of the quasi-Newton method for solving the nonlinear least squares problem where the cost function can be represented as a sum of squared nonlinear functions. In the Gauss-Newton method the proxy for the Hessian is derived from the Jacobian of the cost function. In each iteration of the Gauss-Newton algorithm, functions are locally approximated in a first-order sense and thus the optimization problem boils down to a linear least squares problem. A similar first-order approximation of the squared term in the rhs of Equation 13 leads to

$$\underset{\Delta\mathbf{\Omega}_{\mathcal{V}}}{\text{minimize}} \sum_{(i,j)\in\mathcal{E}} \phi_{ij}.\left\|\mathbf{r}_{ij}\left(\mathbf{0}\right)+\mathbb{J}\mathbf{r}_{ij}\left(\mathbf{0}\right)^{T}\Delta\mathbf{\Omega}_{\mathcal{V}}\right\|^{2} \tag{15}$$

where $\mathbb{J}\mathbf{r}_{ij}\left(\mathbf{0}\right)$ is the Jacobian (with respect to the unknown to be estimated, $\Delta\mathbf{\Omega}_{\mathcal{V}}$) of $r_{ij}\left(\Delta\mathbf{\Omega}_{\mathcal{V}}\right)$ at the current point $\Delta\mathbf{\Omega}_{\mathcal{V}}=\mathbf{0}$. Minimizing the cost in Equation 15 is equivalent to solving the following linear system of equations in a least squares sense

$$\sqrt{\phi_{ij}}\mathbb{J}\mathbf{r}_{ij}\left(\mathbf{0}\right)^{T}\Delta\mathbf{\Omega}_{\mathcal{V}}=-\sqrt{\phi_{ij}}\mathbf{r}_{ij}\left(\mathbf{0}\right)\ \ \forall(i,j)\in\mathcal{E} \tag{16}$$

From Equation 10, clearly, $\mathbf{r}_{ij}\left(\mathbf{0}\right)=\Delta\boldsymbol{\omega}_{ij}$. We have derived the form for $\mathbb{J}\mathbf{r}_{ij}\left(\mathbf{0}\right)$ by differentiating the cost function in the Appendix. Consequently, we have

$$\begin{aligned}\mathbb{J}\mathbf{r}_{ij}\left(\mathbf{0}\right)=&\ \alpha_{ij}\underbrace{\left[\ \cdots+\mathbf{I}\cdots-\mathbf{I}\cdots\ \right]}_{\mathbf{A}_{ij}}\\ &+(1-\alpha_{ij})\underbrace{\left[\ \cdots+\frac{\Delta\boldsymbol{\omega}_{ij}\Delta\boldsymbol{\omega}_{ij}^{T}}{\theta_{ij}^{2}}\cdots-\frac{\Delta\boldsymbol{\omega}_{ij}\Delta\boldsymbol{\omega}_{ij}^{T}}{\theta_{ij}^{2}}\cdots\ \right]}_{\mathbf{B}_{ij}}\\ &+\frac{1}{2}\underbrace{\left[\ \cdots+\left[\Delta\boldsymbol{\omega}_{ij}\right]_{\times}\cdots+\left[\Delta\boldsymbol{\omega}_{ij}\right]_{\times}\cdots\ \right]}_{\mathbf{C}_{ij}}\end{aligned} \tag{17}$$

where $\theta_{ij}=\|\Delta\boldsymbol{\omega}_{ij}\|$ and $\alpha_{ij}=\frac{\theta_{ij}}{2}\cot\left(\frac{\theta_{ij}}{2}\right)$. $\mathbf{A}_{ij}$, $\mathbf{B}_{ij}$ and $\mathbf{C}_{ij}$ are constructed by placing the respective terms as 3×3 blocks in the appropriate locations of i and j respectively.

Thus, the Gauss-Newton problem in Equation 16 becomes

$$\begin{aligned}&\sqrt{\phi_{ij}}\Big(\alpha_{ij}\mathbf{A}_{ij}+(1-\alpha_{ij})\mathbf{B}_{ij}+\frac{1}{2}\mathbf{C}_{ij}\Big)\Delta\mathbf{\Omega}_{\mathcal{V}}\\ =&\ \ -\sqrt{\phi_{ij}}\Delta\boldsymbol{\omega}_{ij}\ \ \forall(i,j)\in\mathcal{E}\end{aligned} \tag{18}$$

which we approximate (as a quasi-Newton form) as

$$\sqrt{\phi_{ij}}\mathbf{A}_{ij}\Delta\mathbf{\Omega}_{\mathcal{V}}=-\sqrt{\phi_{ij}}\Delta\boldsymbol{\omega}_{ij}\ \ \forall(i,j)\in\mathcal{E} \tag{19}$$

Now collecting all the equations for individual edges (i.e. $\forall(i,j)\in\mathcal{E}$) into a single system of equation we get,

$$\sqrt{\Phi}\mathbf{A}\Delta\mathbf{\Omega}_{\mathcal{V}}=-\sqrt{\Phi}\Delta\mathbf{\Omega}_{\mathcal{E}} \tag{20}$$

where $\mathbf{A}$ is the matrix obtained by stacking $\mathbf{A}_{ij}$ as row matrices. Φ is a diagonal matrix with its diagonal elements populated by ϕ_{ij}.

Here, we make a few observations regarding our approximation in Equation 19. Firstly, this approximation provides us some advantages. We note that α_{ij}, θ_{ij}, $\mathbf{B}_{ij}$ and $\mathbf{C}_{ij}$ depend on the current residuals ($\Delta\mathbf{\Omega}_{\mathcal{E}}$) that change with each iteration. Therefore, by eliminating these terms, we eliminate extra computational expenses in each iteration. Further, this approximation leads to a $3M\times3N$ $\mathbf{A}$ matrix which is the Kronecker product between the identity matrix and the edge incidence matrix of the view-graph $\mathcal{G}$. The system of $3M$ equations involving $3N$ unknowns in Equation 20 can be decoupled into 3 systems of equations each having M equations of N unknowns. Each row of $\mathbf{A}$ corresponds to an edge in the view-graph and contains only two non-zero entries one of which is $+1$ and the other is -1. Consequently, $\mathbf{A}$ depends solely on the topology of the view-graph $\mathcal{G}$ and since it does not change with the iterations, it needs to be computed only once. These factors result in a further speedup.

Secondly, we note that the approximation made in Equation 19 is a small angle approximation for θ_{ij}. However, even when θ_{ij} are not small (for example, when we are far from the optimal value or when the measurement noise is high) this is still a valid quasi-Newton optimization which guarantees that the updates are in descent direction. As we shall see in the following sections, this ensures that our method terminates to a stationary point of the underlying cost function where the gradient is zero. We remark here in passing that the joint averaging approach of [6], which is also used in this paper, has been misinterpreted as a first-order approximation of the underlying problem in [24]. The use of an approximation of an intermediate iteration step does not imply that the overall cost function is approximated.

Now solving Equation 20 in a least squares sense is equivalent to minimizing $\left(\sqrt{\Phi}\mathbf{A}\Delta\mathbf{\Omega}_{\mathcal{V}}+\sqrt{\Phi}\Delta\mathbf{\Omega}_{\mathcal{E}}\right)^{2}$. Therefore, our update step can be written as

$$\Delta\mathbf{\Omega}_{\mathcal{V}}=-\left(\mathbf{A}^{T}\Phi\mathbf{A}\right)^{-1}\mathbf{A}^{T}\Phi\Delta\mathbf{\Omega}_{\mathcal{E}} \tag{21}$$

Note that $\left(\mathbf{A}^{T}\Phi\mathbf{A}\right)$ is the weighted Laplacian matrix of the view-graph $\mathcal{G}$ and is often very sparse in practice since only a small fraction of all the possible camera pairs have matching relationships, i.e. the number of edges $|\mathcal{E}|=M$ is much smaller than the maximum possible ${}^{N}C_{2}=\frac{N(N-1)}{2}$ edges. This sparsity further helps us in solving Equation 21 efficiently.

To summarize our approach thus far, we start with a non-linear cost function (Equation 6) that we solve iteratively. We consider the form of an update step in a given iteration that leads to the minimization problem in Equation 9. Setting the gradient of the corresponding cost function to zero leads to Equation 14, which is an IRLS minimization problem, albeit non-linear. Further, this non-linear problem is solved by using the quasi-Newton step of Equation 21 which is of the IRLS form for linear least-squares. We can now summarize our algorithm for robust averaging of relative rotations as given in Algorithm 1.

Algorithm 1 Relative Rotation Averaging with iteratively reweighted least squares

```
Input: Set of relative rotation measurements $\{\mathbf{R}_{ij}\}$ for $ij \in \mathcal{E}$, tolerance $\epsilon$, maximum iterations $N_{max}$
Output: Set of absolute rotations $\mathbf{R}_{\mathcal{V}} = \{\mathbf{R}_1, \cdots, \mathbf{R}_N\}$
Initialisation: Set $\mathbf{R}_{\mathcal{V}} = \{\mathbf{R}_1, \cdots, \mathbf{R}_N\}$ to an initial guess. Set iteration number $k = 1$
    while $\Delta\mathbf{\Omega}_{\mathcal{V}} > \epsilon$ AND $k < N_{max}$ do
        1. $\Delta\boldsymbol{\omega}_{ij} \leftarrow \boldsymbol{\omega}(\mathbf{R}_j^{-1}\mathbf{R}_{ij}\mathbf{R}_i)$
        2. Collect $\Delta\boldsymbol{\omega}_{ij}\ \forall ij \in \mathcal{E}$ into $\Delta\mathbf{\Omega}_{\mathcal{E}}$
        3. $\phi_{ij} \leftarrow \phi\left(r_{ij}\left(\mathbf{0}\right)\right)$ where $r_{ij}$ is defined in Equation 10
        4. Collect $\phi_{ij}\ \forall ij \in \mathcal{E}$ into a diagonal matrix $\Phi$
        5. $\Delta\mathbf{\Omega}_{\mathcal{V}} \leftarrow -\left(\mathbf{A}^T\Phi\mathbf{A}\right)^{-1}\mathbf{A}^T\Phi\Delta\mathbf{\Omega}_{\mathcal{E}}$
        6. $\forall i \in \{1, N\}, \mathbf{R}_i \leftarrow \mathbf{R}\mathbf{R}(\Delta\boldsymbol{\omega}_i)$
        7. $k \leftarrow k + 1$
    end while
```

4.3 Characterizing the Properties of Our Method

In this subsection, we show that the gradient of our cost function (Equation 9) is zero at the point of convergence of Algorithm 1. This proves that our algorithm reaches a stationary point of the underlying cost function. Such a point could be a local minimum or a saddle point of the cost function. [25] discusses the theoretical existence of saddle points for the problem of relative rotation averaging. However, in our extensive real and synthetic experiments, we always found our method to converge very close to the ground truth.

Using Equation 12, we rewrite the gradient of the cost function as

$$\begin{aligned}
&\boldsymbol{\nabla} F\left(\Delta\mathbf{\Omega}_{\mathcal{V}}\right)\\
=\ & \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} \phi\left(\left\|\mathbf{r}_{ij}\left(\Delta\mathbf{\Omega}_{\mathcal{V}}\right)\right\|\right)\boldsymbol{\nabla}\left(\left\|\mathbf{r}_{ij}\left(\Delta\mathbf{\Omega}_{\mathcal{V}}\right)\right\|^2\right)\\
=\ & \sum_{(i,j)\in\mathcal{E}} \phi\left(\left\|\mathbf{r}_{ij}\left(\Delta\mathbf{\Omega}_{\mathcal{V}}\right)\right\|\right)\mathbf{r}_{ij}\left(\Delta\mathbf{\Omega}_{\mathcal{V}}\right)^T\mathbb{J}\left(\mathbf{r}_{ij}\left(\Delta\mathbf{\Omega}_{\mathcal{V}}\right)\right)
\end{aligned} \tag{22}$$

Therefore at a given iteration of the optimization, the gradient is given by

$$\begin{aligned}
&\boldsymbol{\nabla} F\left(\mathbf{0}\right)\\
=\ & \sum_{(i,j)\in\mathcal{E}} \phi\left(\left\|\mathbf{r}_{ij}\left(\mathbf{0}\right)\right\|\right)\mathbf{r}_{ij}\left(\mathbf{0}\right)^T\mathbb{J}\left(\mathbf{r}_{ij}\left(\mathbf{0}\right)\right)\\
=\ & \sum_{(i,j)\in\mathcal{E}} \phi_{ij}\Delta\boldsymbol{\omega}_{ij}^T\left(\alpha_{ij}\mathbf{A}_{ij} + (1-\alpha_{ij})\mathbf{B}_{ij} + \frac{1}{2}\mathbf{C}_{ij}\right)\\
=\ & \sum_{(i,j)\in\mathcal{E}} \phi_{ij}\left(\alpha_{ij}\Delta\boldsymbol{\omega}_{ij}^T\mathbf{A}_{ij} + (1-\alpha_{ij})\Delta\boldsymbol{\omega}_{ij}^T\mathbf{B}_{ij} + \mathbf{0}\right)\\
=\ & \sum_{(i,j)\in\mathcal{E}} \phi_{ij}\left(\alpha_{ij}\Delta\boldsymbol{\omega}_{ij}^T\mathbf{A}_{ij} + (1-\alpha_{ij})\Delta\boldsymbol{\omega}_{ij}^T\mathbf{A}_{ij} + \mathbf{0}\right)\\
=\ & \Delta\mathbf{\Omega}_{\mathcal{E}}{}^T\Phi\mathbf{A}
\end{aligned} \tag{23}$$

In Equation 23, from the form of $\mathbf{C}_{ij}$ in Equation 17, we have $\Delta\boldsymbol{\omega}_{ij}^T\mathbf{C}_{ij} = \mathbf{0}$. Moreover $\Delta\boldsymbol{\omega}_{ij}^T\mathbf{B}_{ij}$ can be seen to reduce to the form of $\Delta\boldsymbol{\omega}_{ij}^T\mathbf{A}_{ij}$. In our algorithm, we stop only when the solution to Equation 20 is $\Delta\mathbf{\Omega}_{\mathcal{V}}^k = \mathbf{0}$, i.e. when the right hand side $(-\sqrt{\Phi}\Delta\mathbf{\Omega}_{\mathcal{E}})$ lies outside the column space of $\sqrt{\Phi}\mathbf{A}$. This is equivalent to the criteria $\left(\sqrt{\Phi}\mathbf{A}\right)^T\left(-\sqrt{\Phi}\Delta\mathbf{\Omega}_{\mathcal{E}}\right) = \mathbf{0}$. Therefore, on termination of our algorithm, $\left(\sqrt{\Phi}\mathbf{A}\right)^T\left(-\sqrt{\Phi}\Delta\mathbf{\Omega}_{\mathcal{E}}\right) = \mathbf{A}^T\Phi\Delta\mathbf{\Omega}_{\mathcal{E}} = \boldsymbol{\nabla} F\left(\mathbf{0}\right)^T = \mathbf{0}$. Therefore, our algorithm terminates to a stationary point of the underlying cost function. We may now investigate this property of our method by considering two special cases.

Case 1: Assume that we are using a least squares or quadratic loss function i.e. $\rho(x) = x^2$. Therefore, Φ is an identity matrix. On termination of our algorithm, $\mathbf{A}^T\left(\Delta\mathbf{\Omega}_{\mathcal{E}}\right) = \mathbf{0}$. Recall that, $\mathbf{A}$ is the edge incidence matrix of the view-graph, where columns correspond to the vertices and rows correspond to edges. Consider, the j-th $N \times 3$ block of $\mathbf{A}$. This block signifies the edges that are connected to the j-th camera. Therefore, the fact that $\mathbf{A}^T\left(\Delta\mathbf{\Omega}_{\mathcal{E}}\right) = \mathbf{0}$ implies $\sum_{i\in\mathcal{N}(j)} \boldsymbol{\omega}(\mathbf{R}_j^{-1}\mathbf{R}_{ij}\mathbf{R}_i) = \mathbf{0}$, where $\mathcal{N}(j)$ is the set of vertices connected to i. Now, $\mathbf{R}_{ij}\mathbf{R}_i$ is the position of j-th camera estimated from the connecting edge from camera i. Therefore, this shows that vertex j is at the geodesic mean of all the estimates suggested by its neighbors. This proves that on termination of our method, each and every camera $i \in \mathcal{V}$ is oriented along the geodesic or intrinsic mean of the rotations suggested by the edges incident on camera i. In fact this is the principle also used for the distributed least squares rotation averaging in [25].

Case 2: Assume that we are using an ℓ_1 loss function i.e. $\rho(x) = |x|$. A similar argument as in case 1 proves that on termination of our method, each and every camera $i \in \mathcal{V}$ is oriented along the geodesic or intrinsic median of the rotations suggested by the edges incident on camera i. Once again, this is the principle used for the distributed ℓ_1 rotation averaging in [13]. However, the difference of these approaches compared to ours lies in the method used for optimization.

4.4 About the Nature and Convergence of Our Method

In the previous subsection, we have shown that our method terminates to a stationary point of the underlying cost function. In this subsection, we show that in each iteration of Algorithm 1 we choose a descent direction to update our estimate. Referring to Equation 21 and 23, our update steps can be written as

$$\Delta\mathbf{\Omega}_{\mathcal{V}} = -\left(\mathbf{A}^T\Phi\mathbf{A}\right)^{-1}\boldsymbol{\nabla} F\left(\mathbf{0}\right)^T \tag{24}$$

Since $\left(\mathbf{A}^T\Phi\mathbf{A}\right)$ is a positive semi-definite matrix our update steps are always along a descent direction. Note that, this property by itself does not guarantee that the cost is minimized at every iteration as the size of the update step may be larger than what is required to reach the directional minimum. This can be remedied by performing a line search along the update direction. However, in our extensive experiments, we have never needed such a computationally expensive line search, and our approach always converged very close to the ground truth.

4.5 Initialization and Further Speed Improvements

As discussed earlier, while loss functions like Geman-McClure or $\ell_{\frac{1}{2}}$ are robust to outliers, they are also prone to converge to local minima. Therefore, we initialize our method by first optimizing an ℓ_1 loss function for a few iterations (5 in our experiments) and then we minimize a more robust loss function. We parenthetically note here that in our implementation, the ℓ_1 minimization is initialized using the solution obtained using a random spanning tree of the view-graph.

However, carrying out ℓ_1 minimization using the IRLS method is time consuming since we need to compute weights in each iteration of Algorithm 1 to minimize $\|\Phi(\mathbf{A}\Delta\mathbf{\Omega}_{\mathcal{V}} + \Delta\mathbf{\Omega}_{\mathcal{E}})\|^2$. In any event, for us this ℓ_1 minimization is only a way to obtain a reasonable initial estimate. Therefore, to reduce the computation cost, we choose to minimize $\|(\mathbf{A}\Delta\mathbf{\Omega}_{\mathcal{V}} + \Delta\mathbf{\Omega}_{\mathcal{E}})\|$ instead of $\|\Phi(\mathbf{A}\Delta\mathbf{\Omega}_{\mathcal{V}} + \Delta\mathbf{\Omega}_{\mathcal{E}})\|^2$. We can efficiently minimize $\|(\mathbf{A}\Delta\mathbf{\Omega}_{\mathcal{V}} + \Delta\mathbf{\Omega}_{\mathcal{E}})\|$ using a convex programming approach [12], [26]. This approach is faster than computing weights in every iteration and is also robust to outliers. Therefore, we choose to minimize $\|(\mathbf{A}\Delta\mathbf{\Omega}_{\mathcal{V}} + \Delta\mathbf{\Omega}_{\mathcal{E}})\|$ which we shall refer to as the 'initial ℓ_1' cost for the first few iterations. Thereafter, we start the optimization of the desired cost function considering the appropriate weight matrix. Our initial ℓ_1 optimization is described in Algorithm 2 and our overall algorithm is summarized in Algorithm 3.

Algorithm 2 Initial ℓ_1 Relative Rotation Averaging

```
Input: Set of relative rotation measurements $\{\mathbf{R}_{ij}\}$ for $ij \in \mathcal{E}$
Output: Set of absolute rotations $\mathbf{R}_{\mathcal{V}} = \{\mathbf{R}_1, \cdots, \mathbf{R}_N\}$
Initialisation: Set iteration $k = 1$ and $\mathbf{R}_{\mathcal{V}}$ to any initial value
    while $\Delta\mathbf{\Omega}_{\mathcal{V}} > \epsilon$ AND $k < N_{max}$ do
        1. $\Delta\boldsymbol{\omega}_{ij} = \boldsymbol{\omega}(\mathbf{R}_j^{-1}\mathbf{R}_{ij}\mathbf{R}_i)$
        2. Collect $\Delta\boldsymbol{\omega}_{ij} \forall (i,j) \in \mathcal{E}$ into $\Delta\mathbf{\Omega}_{\mathcal{E}}$
        3. $\Delta\mathbf{\Omega}_{\mathcal{V}} = \underset{\Delta\mathbf{\Omega}_{\mathcal{V}}}{\text{argmin}} \|\mathbf{A}\Delta\mathbf{\Omega}_{\mathcal{V}} + \Delta\mathbf{\Omega}_{\mathcal{E}}\|$
        4. $\forall i \in \{1, N\}, \mathbf{R}_i \leftarrow \mathbf{R}_i\mathbf{R}(\Delta\boldsymbol{\omega}_i)$
        5. $k \leftarrow k + 1$
    end while
```

Algorithm 3 Complete Relative Rotation Averaging Algorithm

Initial ℓ_1 step:

- Initialise $\mathbf{R}_{\mathcal{V}}$ to initial guess
- Run 5 iterations of Algorithm 2

Iteratively Reweighted Least Squares step:

- Set $\mathbf{R}_{\mathcal{V}}$ to output of Initial ℓ_1 step
- Run Algorithm 2 until convergence

5 Existing Methods

In this section, we briefly discuss the other existing methods for relative rotation averaging with a focus on two state-of-the art approaches that we compare our method with, i.e. the Weiszfeld approach of [13] and the discrete-continuous optimization (DISCO) approach of [11]. The problem of relative rotation averaging was first introduced in [8] where a linear solution is obtained using a quaternion representation for rotations under

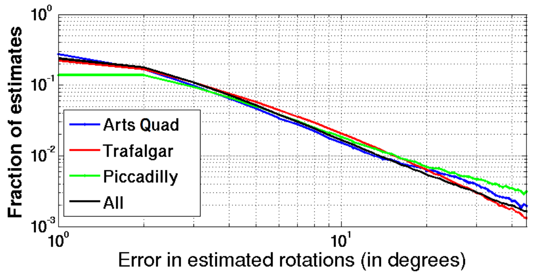


Fig. 2. Log-log plot of distribution of errors in relative rotations for various large-scale datasets. The almost linear behaviour of all datasets implies that it is not possible to choose a threshold for removing outliers. See text for details.

a small noise assumption. Since quaternions $\mathbf{q}$ and $-\mathbf{q}$ both represent the same rotation, this approach needs to fix this sign ambiquity which makes the problem more complicated. In [6], the problem of motion averaging is solved using a Lie group representation. Both of these methods do not incorporate robustness to outliers in the observations. Subsequent robust approaches can be classified according to the manner in which they handle outliers. The first set of approaches [11], [27], [28] identify outliers and remove them before proceeding with the averaging step. The second set of methods that include [12], [13] solve the problem by considering all observations and suppressing the influence of outliers.

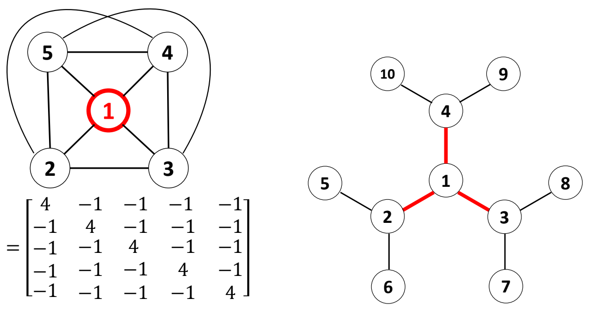
Methods that remove outliers : The non-robustness of [6] was addressed in [27] which used a RANSAC approach to detect and remove outlier edges from the viewgraph. Once outliers are removed, the ℓ_2-based averaging approach of [6] is applied. The outlier removal technique of [28] relies on the fact that composition of transformations in a loop should result in an identity transformation. In [28] statistics are collected on many loops of the viewgraph. This information is fed into a belief propagation formulation to identify outliers. Both [27], [28] are extremely computationally costly for large-scale problems. Recently, [11] proposed the DISCO algorithm which also uses a belief propagation method for outlier detection. In [11] the problem of relative rotation averaging is solved in two steps. In the first step, by ignoring the twist component of camera rotations, the parameter space of 3D rotations is coarsely discretized into a small set of discrete rotations. The problem of robust averaging is then performed using loopy belief propagation on a label set corresponding to the discrete rotations. The approximate solution of this step is used to detect and remove outliers. Subsequently, DISCO refines this discrete solution using a non-linear optimization based on the Rodrigues parameters of rotations. While we will characterize the performance of DISCO in Section 6, we remark here that DISCO converts the problem of robust geometric estimation into one of discrete optimization using belief propagation. Thus, while DISCO can handle large-scale problems, it does so at a very high computational cost and requires expensive hardware in the form of a cluster. Additionally, DISCO fails to exploit the geometric structure of $SO(3)$.

Apart from their high computational cost, the above methods that identify outliers are unsatisfactory for another reason. Any approach that identifies and removes outliers requires the specification of a threshold that is used for classification. While even under favorable circumstances specifying such a threshold is a bit of a black art, in our context of rotation averaging for SfM problems it is often infeasible to define this threshold in a principled fashion. Consider Figure 2 which shows the distribution of errors in the set of relative rotations $\mathbf{R}_{\mathcal{E}}$, i.e. the distribution of $\{d(\mathbf{R}_{ij}, \mathbf{R}_j^G(\mathbf{R}_i^G)^{-1}) | \forall (i,j) \in \mathcal{E}\}$ where $\mathbf{R}_k^G$ is the ground truth for the k-th camera. This distribution of errors is shown for some large-scale datasets as well as for all datasets considered together (marked 'All'). From the linear nature of the plot on a log-log scale, we can easily infer that it is not possible to specify an obvious outlier threshold. In other words, for real-world data it is not possible to clearly classify relative rotations as inliers or outliers, hence the need to incorporate robustness in the averaging step without removing outliers.[1]

Methods that estimate in the presence of outliers : The Weiszfeld method of [13] addresses the problem of robust estimation in the presence of outlier observations. Working with the Lie group structure of $SO(3)$, this approach incorporates robustness by replacing the ℓ_2 distance with the robust ℓ_1 distance in the computations in the Lie algebra. While this allows for robustness, the estimate of [13] is a distributed averaging approach [29], [30] where each individual vertex (camera) in the viewgraph $\mathcal{G}$ is updated in turn, while holding the rotation estimates of the rest of the cameras fixed. An equivalent distributed approach of rotation averaging for outlier free cases can be found in [25]. The rate of convergence of [13] is extremely slow as the vertices are updated one at a time. In addition, we show example scenarios where the Weiszfeld update of [13] fails resulting in a completely wrong solution. In [31], a distributed averaging technique is further utilized for a general ℓ_q norm optimization. Like the Weiszfeld method, the approach in [32] is also a distributed approach to relative rotation averaging. [32] provides a guarantee of global convergence with a specific choice of loss function under perfect noise-free measurements and the experiments are for small datasets comprising at most 124 cameras. However, under noise-free measurements, since we are not restricted to distributed estimation, we can simply use a spanning tree of the view-graph to estimate all the unknown rotations perfectly. In other words, solving the noise-free problem is trivial for us. In [33], a quaternion representation of rotation is used to solve the problem of relative rotation averaging. The RANSAC approach of [27] is used to fix the sign ambiguity of quaternions. Subsequently, using a quaternion representation, [33] refines its solution using an IRLS approach and provides for a test for global minima using Lagrangian duality. In [34], the theory of sparse matrix recovery and completion is applied using an orthonormal matrix representation of 3D rotations. Apart from having to relax the orthonormality constraints of rotation matrices, their approach uses a huge matrix to represent the rotations between all pairs of cameras, which makes their approach infeasible for large-scale problems.

The approach of this paper belongs to the later category of methods that work in the presence of outliers. Like [6], [12], it utilizes the Lie group structure of $SO(3)$ but also incorporates robustness without sacrificing either speed or accuracy. In contrast to the distributed approach of [13], our method jointly solves for all unknown 3D rotations.

1. For the plots in Figure 2 since we do not have a true ground truth $\mathbf{R}_j\mathbf{R}_i^{-1}$, we have used an incremental BA solution as our 'ground truth'. However this does not change our basic inference that no clear outlier threshold exists for real data.



$$A^T A = \begin{bmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & -1 & 4 \end{bmatrix}$$

(a) Slow convergence (b) Spanning Tree (failure case)

Fig. 3. Synthetic graphs to demonstrate problems of slow convergence and failure of the Weiszfeld method of [13]. Please see text for details.

Comparison with the Weiszfeld method [13]: While the Weiszfeld method of [13] also solves the problem on the $SO(3)$ group, it suffers from the twin problems of slow convergence and failure cases. We illustrate both these outcomes using the following two synthetic experiments.

Slow Convergence : Consider the view graph in Figure 3(a) consisting of 5 vertices that form a full graph of all possible edges. Suppose that the vertices represent rotations in the XY plane, i.e. the rotations have only one degree of freedom making this a one-dimensional problem. Let all vertices be assigned to the ground truth values except for vertex 1 which is set to a wrong value. Therefore, if we update only vertex 1 we can reach the optimal, ground truth configuration. At this point the gradient of the cost function $\boldsymbol{\nabla} F$ is proportional to $\begin{bmatrix} 4 & -1 & -1 & -1 & -1 \end{bmatrix}^T$ and following a distributed averaging method starting with vertex 1, we can immediately reach the correct optima. However, if we start our updates at any other vertex, then these vertices will also be moved from their correct position which in turn will take many iterations to converge to the optima. In contrast, in our joint averaging scheme, the update step is computed as $(\mathbf{A}^T\mathbf{A})^{-1}\boldsymbol{\nabla} F = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T$ which results in only one vertex being moved, resulting in faster and quicker convergence. This observation can be generalized to larger graphs with chunks of correct estimates, where the distributed averaging approach of [13] will take a long time to converge.

Failure Case : We illustrate the possibility of the failure of the Weiszfeld method of [13] using a scenario as follows. Consider the situation depicted in Figure 3(b) where we have a graph of 10 vertices. We assume that we have a full graph here of 45 relative rotation estimates. For ease of visualization, we only depict only a specific spanning tree of interest here. We note that the method of [13] uses the rotations estimated using a spanning tree as the intial guess for its distributed averaging iteration. Let us assume that the relative rotation estimates for the three thick red edges are outliers whereas all other edges have perfect measurements. Under this scenario, as is done in [13], let us generate an initial guess for the rotation estimates using this spanning tree. Note that this spanning tree contains 3 outliers which divides the graph into 3 chunks such that the initial rotation estimates are grossly wrong (although each individual chunk is correct). Now each node has

9 neighbors (since we have a full graph) such that the estimated rotations from the neighbors will have 3 different values. We synthesized such situations with the ground truth rotations for the 10 vertices generated randomly. For more than 70% of the time, the distributed rotation averaging approach of [13] converged to a grossly wrong minima, whereas our joint optimization technique always converged to the correct estimate.

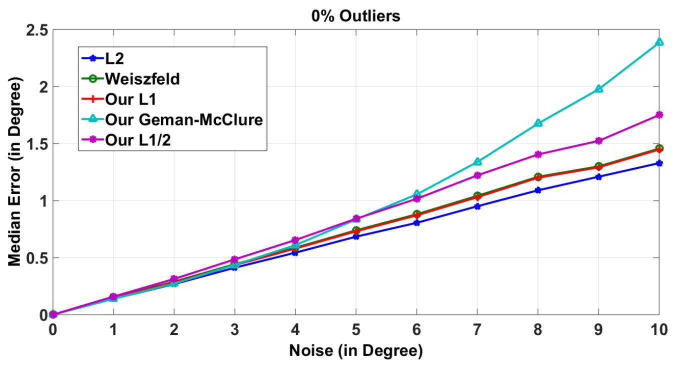
6 Results

In this section, we demonstrate the efficiency and accuracy of our relative rotation averaging approach on some large-scale real-world data sets. We begin with a description of the data sets used.

Data set characteristics: In Table 2, we summarize the characteristics of the real world data sets we have used in our experiments. For dataset ‘Notre Dame 715’ (ND2) we used the raw images available online[2] to compute the pairwise relative rotations from the two-image BA solution obtained using the `bundler` [2] software. For the ‘Arts Quad’ (ARQ) and ‘San Francisco’ (SNF) data sets, the relative rotations $\mathbf{R}_{ij}$ have been provided by the authors of [11][3]. For other datasets used, both the raw images as well as estimated relative rotations are provided by the authors of [9][4]. All these data sets come with an estimate obtained using incremental BA which we have used as the ground truth in our experiments. The number of cameras in the largest connected component of these data sets is indicated in the second column ‘# Camera’. The third column ‘# Edge’ indicate the number of relative rotation estimates available within these connected components. However, the ground truth results are not available for all the connected cameras. The number of cameras for which the ground truth is available is indicated under the column ’# Ground truth’. Only these cameras for which the ground truth is available can be used to evaluate the accuracy of any algorithm. We note here that while we ran both our method and the Weiszfeld method of [13] on all the cameras in the largest connected component, the method of DISCO [11] first eliminates some of the cameras and edges from a connected component and therefore solves for a much smaller number of cameras compared to us. We have evaluated the level of data corruption present in the relative rotation estimates of different data sets. To do so, we have estimated the relative rotations along the connecting edges from the known ground truth and compared these values to the measured relative rotations. The mean, median and RMS error obtained in this manner is also indicated in Table 2. We also have tabulated the percentage of edges having errors greater than $10°$, $30°$, $60°$ and $90°$. Taken together, these statistics provide information about the amount of noise and outliers present in individual data sets. For example, of the data sets considered, we see that ‘San Francisco’ (SNF) is the cleanest data set, while ‘Madrid Metropolis’ (MDR) is the most corrupted data set.

Different robust loss functions: We have experimented with a wide variety of well-known robust loss functions to solve the problem of relative rotation averaging. We have chosen the parameter α (please see Table 1) to be equal to $5°$ with appropriate tuning parameters [23] wherever applicable. In Table 3, we show the median errors for these different loss functions tested on large-scale real-world data sets by running Algorithm 3. We also mark the best result for each of the data sets in bold. The computation times for these loss functions are tabulated in Table 4. We find that $\ell_{\frac{1}{2}}$ and Geman-McClure consistently yield smaller median error compared to the other lost functions, while there is no significant difference in computational time amongst the different loss functions.

2. http://phototour.cs.washington.edu/datasets
3. http://vision.soic.indiana.edu/disco
4. http://www.cs.cornell.edu/projects/1dsfm/



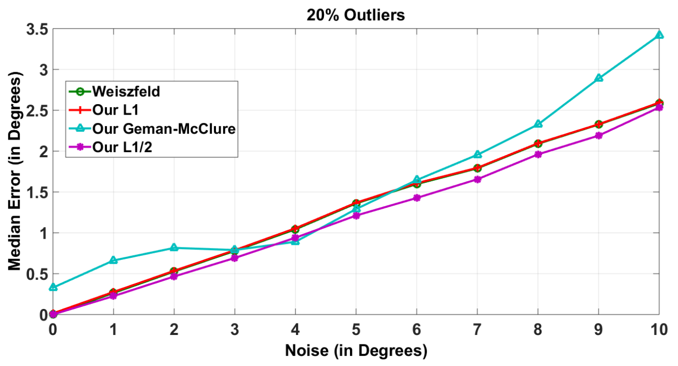


Fig. 4. Testing the statistical efficiency of different loss functions on synthetic datasets. See text for details.

Statistical efficiency: We have also tested for the statistical efficiency of different robust loss functions on synthetic datasets. Since both $\ell_{\frac{1}{2}}$ and Geman-McClure provide similar performance on real datasets, our experiment on the synthetic dataset is designed to help choose between them. For this experiment, we chose the number of cameras to be 100 where the ground truth rotations are picked randomly. Relative rotations are generated from these ground truth rotations and are perturbed with different amount of noise. In this experiment, only 50% of relative rotations or edges in the view-graph are used and all results are averaged over 25 trials. Figure 4(a) shows the performance of different loss functions at different noise levels (0 to 10 degrees) when no outliers are present. Expectedly, the ℓ_2 loss function provides the best result whereas our ℓ_1 method and the Weiszfeld algorithm of [13] give the same results as they minimize the same loss functions. These results are slightly poorer than ℓ_2 loss function and are closest to the quality of the ℓ_2 loss function, i.e. ℓ_1 has the best statistical efficiency. While $\ell_{\frac{1}{2}}$ has poorer statistical efficiency, the Geman-McClure loss function performs even worse. Importantly, it will be noted that the performance of the Geman-McClure loss function starts to degrade significantly beyond the noise level of $5°$. This is because the parameter α of the Geman-McClure loss function was chosen to be $5°$. Therefore, when the noise level is lower than $5°$, virtually all edges are treated as inliers and averaging is done over all edges, although not in an optimal fashion. But when the noise level is greater than

Dataset	#	#	# Ground	Ground truth fitting error						
	Camera	Edge	truth	Mean	Median	RMS	$\% > 10°$	$\% > 30°$	$\% > 60°$	$\% > 90°$
Ellis Island (ELS)	247	20297	227	12.50	2.89	27.43	27.5	12.3	4.6	2.0
Piazza Del Popolo (PDP)	354	24710	338	8.38	1.81	21.50	15.3	8.0	3.8	1.6
NYC Library (NYC)	376	20680	332	14.14	4.22	28.57	31.8	13.6	6.0	2.7
Madrid Metropolis (MDR)	394	23784	341	29.30	9.34	51.45	48.8	29.0	16.6	10.4
Yorkminster (YKM)	458	27729	437	11.16	2.68	27.42	19.2	9.6	5.2	3.0
Montreal Notre Dame (MND)	474	52424	450	7.51	1.67	21.26	13.0	6.3	3.0	1.6
Tower of London (TOL)	508	23863	472	11.58	2.60	28.28	19.9	10.3	5.5	3.2
Notre Dame (ND1)	553	103932	553	14.15	2.70	33.48	22.7	12.9	7.6	4.4
Alamo (ALM)	627	97206	577	9.09	2.78	21.73	17.9	7.3	3.3	1.7
Notre Dame (ND2)	715	64678	715	3.58	1.48	8.20	7.8	1.3	0.3	0.1
Vienna Cathedral (VNC)	918	103550	836	11.26	2.59	27.54	20.7	9.4	5.2	2.9
Union Square (USQ)	930	25561	789	9.02	3.61	19.23	22.8	5.8	2.2	1.1
Roman Forum (ROF)	1134	70187	1084	13.83	2.97	31.85	23.7	12.6	7.1	4.1
Piccadilly (PIC)	2508	319257	2152	19.09	4.93	37.40	35.9	19.3	9.9	5.2
Trafalgar (TFG)	5433	680012	5058	8.62	3.01	18.75	21.3	6.5	2.2	0.9
Arts Quad (ARQ)	5530	222044	4978	9.23	2.49	19.76	22.5	8.7	2.7	0.8
San Francisco (SNF)	7866	101512	7866	1.80	0.99	3.93	1.6	0.3	0.1	0.0

TABLE 2
Datasets and their accuracies.

Data set	Median Errors (in degrees)												
	ℓ_2	ℓ_1	$\ell_{\frac{1}{2}}$	Geman McClure	Huber	Pseudo Huber	Andrews	Bisquare	Cauchy	Fair	Logistic	Talwar	Welsch
ELS	3.91	1.35	1.15	**1.08**	1.75	1.68	1.37	1.30	1.47	1.86	1.73	1.37	1.39
PDP	9.93	4.64	2.62	**2.16**	4.97	5.17	2.32	2.31	2.99	5.90	5.11	2.11	2.42
NYC	7.17	2.07	**1.40**	1.43	3.01	2.68	1.87	1.89	2.32	3.04	2.82	2.19	1.95
MDR	11.40	4.25	**3.08**	4.52	5.12	5.12	6.20	6.26	4.44	6.02	5.17	5.41	6.47
YKM	8.04	1.85	**1.62**	1.70	2.64	2.35	1.95	1.93	2.12	2.62	2.42	2.06	1.94
MND	4.32	0.93	**0.71**	0.77	1.33	1.26	0.87	0.89	1.04	1.64	1.29	0.90	0.98
TOL	5.10	2.74	**2.45**	2.59	3.07	2.95	2.83	2.84	2.95	3.07	3.00	2.97	2.89
ND1	5.27	1.07	**0.98**	1.03	1.47	1.30	1.20	1.24	1.15	1.52	1.35	1.14	1.13
ALM	5.42	2.47	2.14	**2.12**	2.86	2.74	2.26	2.26	2.27	3.05	2.76	2.27	2.37
ND2	1.31	0.50	**0.49**	0.54	0.79	0.71	0.66	0.67	0.69	0.78	0.75	0.73	0.67
VNC	17.28	4.73	**4.64**	4.87	5.32	5.06	5.00	5.13	5.05	5.55	5.14	4.98	5.20
USQ	11.22	**4.93**	4.97	**4.93**	6.20	5.93	5.28	5.32	5.30	6.61	6.05	5.61	5.27
ROF	17.91	2.50	1.70	**1.62**	3.71	3.41	1.92	1.93	2.32	4.66	3.53	2.00	1.95
PIC	27.93	7.09	**3.12**	4.14	8.88	9.10	5.57	5.74	4.61	11.65	9.14	7.03	4.98
TFG	5.36	2.41	2.03	**1.94**	3.03	2.86	2.39	2.40	2.79	3.09	2.93	2.40	2.47
ARQ	10.38	3.36	2.54	**1.97**	4.48	4.10	2.44	2.45	3.44	4.88	4.23	2.46	2.85
SNF	6.63	3.64	3.56	3.05	5.12	4.62	4.40	4.40	4.49	4.84	4.85	4.83	4.38

TABLE 3
Errors (degree) of different robust cost functions.

Data set	Computational time (in seconds)												
	ℓ_2	ℓ_1	$\ell_{\frac{1}{2}}$	Geman McClure	Huber	Pseudo Huber	Andrews	Bisquare	Cauchy	Fair	Logistic	Talwar	Welsch
ELS	1	2	3	2	2	2	2	2	2	2	2	2	2
PDP	1	3	4	4	3	3	3	2	3	2	2	2	2
NYC	1	2	3	2	2	2	2	2	2	2	2	2	2
MDR	2	3	4	6	3	3	4	4	4	2	3	3	4
YKM	1	2	3	2	2	2	2	2	2	2	2	2	2
MND	4	9	10	8	8	8	6	6	7	8	8	6	6
TOL	1	2	3	2	2	2	2	2	2	2	2	2	2
ND1	18	27	32	32	24	24	27	27	21	22	24	27	24
ALM	10	26	33	31	21	21	26	26	23	19	21	24	26
ND2	4	9	13	10	6	7	10	9	9	6	7	7	8
VNC	19	30	41	70	25	25	23	23	27	25	25	18	27
USQ	2	3	8	13	3	3	5	5	6	3	3	2	5
ROF	12	13	17	12	10	10	10	10	10	10	10	9	9
PIC	168	515	620	1013	495	506	1196	1118	440	580	480	521	1749
TFG	715	864	1184	922	792	860	796	788	855	854	854	785	717
ARQ	151	175	208	232	154	159	204	203	164	154	154	204	190
SNF	289	300	294	300	296	296	306	305	293	294	294	292	290

TABLE 4
Computational time (second) of different robust cost functions.

Data set	Median Error (degree)					Iterations				Computation Time (second)			
	DISCO [11]		Weiszfeld [13]	Our		DISCO [11]	Weiszfeld [13]	Our		DISCO [11]	Weiszfeld [13]	Our	
	BP	BP+ℓ_2		Initial ℓ_1	$\ell_{\frac{1}{2}}$			Initial ℓ_1	$\ell_{\frac{1}{2}}$			Initial ℓ_1	$\ell_{\frac{1}{2}}$
ELS	5.54	1.82	1.66	1.86	**1.15**	20	154	12	5+19	470	21	1	3
PDP	12.11	5.25	3.35	5.12	**2.62**	20	90	12	5+19	583	17	1	4
NYC	9.12	2.59	2.43	3.03	**1.40**	20	331	10	5+15	446	63	1	3
MDR	12.12	6.64	4.37	5.95	**3.08**	20	149	19	5+19	560	30	2	4
YKM	26.17	2.34	2.73	2.53	**1.62**	20	66	11	5+12	641	16	1	3
MND	6.81	1.03	0.92	1.40	**0.71**	20	37	9	5+11	1608	10	2	10
TOL	10.38	2.89	2.73	3.14	**2.45**	20	136	12	5+17	479	34	1	3
ND1	7.48	1.31	1.04	1.53	**0.98**	20	63	14	5+11	4070	24	5	32
ALM	7.86	4.21	3.57	2.72	**2.14**	20	137	11	5+13	3917	55	4	33
ND2	—	—	0.50	0.76	**0.49**	—	63	7	5+10	—	27	2	13
VNC	22.35	14.57	5.14	5.45	**4.64**	20	405	16	5+17	4085	222	8	41
USQ	26.27	7.50	13.54	5.92	**4.97**	20	498	18	5+53	466	221	4	8
ROF	35.36	13.69	2.11	3.62	**1.70**	20	205	16	5+17	1559	121	8	17
PIC	36.00	14.66	7.65	10.46	**3.12**	20	1055	39	5+28	15604	1635	156	620
TFG	91.02	91.62	13.20	3.03	**2.03**	20	1492	23	5+16	43616	5128	541	1184
ARQ	87.12	88.58	6.95	4.19	**2.54**	20	1271	20	5+20	5227	5707	437	208
SNF	54.38	3.61	15.85	4.35	**3.56**	20	881	11	5+13	1413	3186	632	294

TABLE 5
Comparison of our methods with the state-of-the-art DISCO [11] and Weiszfeld [13] methods. In this table, to obtain the results of DISCO we have not used any prior knowledge. If priors are used then the median error of DISCO for Arts Quad dataset comes down to $10.08°$ and $4.77°$ before and after the ℓ_2 optimization respectively. DISCO [11] failed on the 'Notre Dame 715' (ND2) dataset which is indicated as '—'.

$5°$, some of the edges are treated as outliers and their influence in the averaging is reduced. Therefore, the averaging is done on a smaller number of edges resulting in poorer performance. This experiment suggests that the use of the Geman-McClure loss function requires appropriate knowledge of the noise level of the data, whereas the use of $\ell_{\frac{1}{2}}$ requires no such knowledge.

In Figure 4(b) we show the performance of different loss functions for different noise levels in the presence of 20% of outliers. In this case the ℓ_2 loss function has the worst performance as it is not robust to outliers. We have not plotted the curve for the ℓ_2 loss function in Figure 4(b) since its error is significantly greater than the error range of the other loss functions. In the presence of outliers, clearly $\ell_{\frac{1}{2}}$ has the best performance among all the loss functions. Therefore, as we do not know the true noise level in real datasets, we recommend the use of the $\ell_{\frac{1}{2}}$ loss function for robust averaging of relative rotations.

Comparison with state of the art methods: In Table 5, we compare the speed and accuracy of our method to the state of the art DISCO [11] and Weiszfeld [13] methods. Although our framework is generic enough to incorporate a variety of cost functions, we have chosen $\ell_{\frac{1}{2}}$ to compare with other state of the art methods, because, as already seeen, $\ell_{\frac{1}{2}}$ performs better than other cost functions. Clearly, our method ($\ell_{\frac{1}{2}}$) outperforms both the state-of-the-art methods in terms of accuracy and speed.

Accuracy: In Table 5, we have marked the lowest median errors in bold. Our $\ell_{\frac{1}{2}}$ method consistently gives lower median error than the other two methods. Note that DISCO solves the problem of relative rotation averaging in two phases: discrete belief propagation (BP) based optimization using viewing directions, and continuous non-linear optimization using Rodrigues parameters. While the code for the BP optimization is available from the authors (http://vision.soic.indiana.edu/projects/disco/), the code for non-linear optimization is not available. We implemented this non-linear optimization using the *lsqnonlin* routine of MATLAB as specified in [35]. However, in our experience, this optimization is found to be unsatisfactory. For a relatively small dataset like 'Yorkminster' (YKM) it took more than 3 hours to converge. On most other data sets it either did not converge or reported an 'out of memory' error when tested on a desktop with 16GB ram. Therefore, we chose an alternative approach for evaluating DISCO. We first perform the belief propagation optimization using the code provided by the authors of [35] with the default parameters. Guided by this estimate we remove outliers using a threshold of $20°$ as stated in [35]. Finally, instead of performing the non-linear optimization using the *lsqnonlin* routine, we have used our proposed algorithm for relative rotation averaging with an ℓ_2 loss function. The result is tabulated under the column BP+ℓ_2. We remark here that using our ℓ_2 optimization instead of the continuous optimization described in [11] is a favorable alteration to the method of [11]. Note that since ℓ_2 cost function is not robust to outliers, the final accuracy depends on how well the belief propagation optimization of DISCO detects the outliers. We see that for the first nine data sets in Table 2, i.e. 'Ellis Island' (ELS) through 'Alamo' (ALM), the belief propagation optimization of DISCO estimates the rotations reasonably well which in turn helps the ℓ_2 optimization to result in small to moderate median errors for the combined method 'BP + ℓ_2'. For other datasets, DISCO does not perform well. While our test for 'Notre Dame 715' (ND2) crashed, for the datasets from 'Vienna Cathedral' (VNC) through 'San Francisco' (SNF), the belief propagation optimization performed poorly. All of these results are obtained without using any priors in the belief propagation step of DISCO. While incorporating a prior for rotations does improve performance (a median error of $10.08°$ for the Arts Quad (ARQ) data set), such priors are not generally available.

From the performance of DISCO [11] in Table 5, we can draw two important conclusions. The approach of ignoring the inherent geometric structure of $SO(3)$ and converting the robust rotation averaging problem into one of label assignment through belief propagation has significant drawbacks. Apart from demanding large computational resources and a significant computational

Data	ELS	PDP	NYC	MDR	YKM	MND	TOL	ND1	ALM	ND2	VNC	USQ	ROF	PIC	TFG	ARQ	SNF
Sum of squares alignment	1.15	2.62	1.40	3.08	1.62	0.71	2.45	0.98	2.14	0.49	4.64	4.97	1.7	3.12	2.03	2.54	3.56
Absolute sum alignment	0.52	0.90	1.37	1.31	1.59	0.52	2.44	0.65	1.07	0.42	1.27	3.93	1.6	2.94	2.02	2.53	3.31

TABLE 6
Median error metric for different alignment methods. See text for details.

time, DISCO does not work well as the data set grows in size. Moreover, even for smaller datasets, the fact that the median errors for 'BP + ℓ_2' is always worse than that of our $\ell_{\frac{1}{2}}$ approach is further validation of the point made in Section 5 that for real data, we cannot truly separate inliers and outliers. In contrast to DISCO, the Weiszfeld method of [13] does utilize the geometric structure of $SO(3)$ and also sets up an appropriate cost function to be minimized. However, the Weiszfeld method suffers from its use of a distributed averaging approach. Apart from our earlier discussion of the limitations of the Weiszfeld method, in Table 5, we can see that the Weiszfeld method performs reasonably well for smaller data sets, although it is always poorer in performance when compared to our approach. However, as the data set grows in size, the Weiszfeld method deteriorates in performance owing to its slow convergence which is reflected in both higher median error values as well as large computational times required. In all cases, we can note that our approach of using a batch estimation with an $\ell_{\frac{1}{2}}$ loss function provides very good quality estimates in an efficient manner.

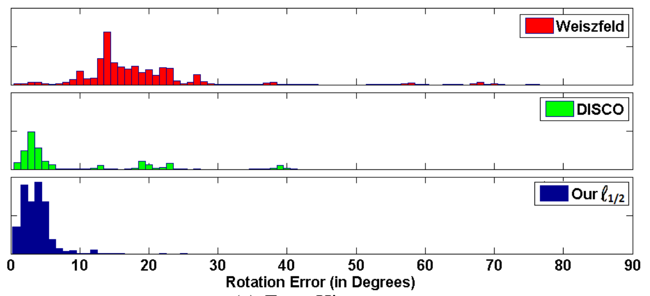
Note that while both our method with an ℓ_1 loss function and the Weiszfeld method optimize the same cost function, results of our method with the ℓ_1 loss function (column ℓ_1 of Table 3) and results of the Weiszfeld method (column 'Weiszfeld' of Table 5 are different. This difference arises due to the optimization approaches leading to different termination criteria. On the one hand, we terminate our method when the average update of all camera vertices is less than 0.001 radians. On the other hand, since the Weiszfeld method has poor convergence behavior, we allow it a more relaxed termination criterion of the update for all vertices being less than 0.001 radians. The Weiszfeld method would perform even more poorly if we were to use the same termination criterion as our approach. However, even with this relaxed criterion, Weiszfeld method often fails to reach very close to the optima. Thus, to go closer to the optima we need to choose an even smaller termination threshold. But smaller termination thresholds increase the accuracy of Weiszfeld method slightly at the cost of a huge increase of computation time. Therefore, the chosen threshold for the Weiszfeld method is the one for which Weiszfeld method produces accurate enough result within reasonable time. Further, the above issue is an intrinsic problem of any distributed averaging. Such averaging often reaches a solution when individual update of any camera parameter do not yield any significant improvement of the cost function. Thus, such methods prematurely terminate in those configurations. However, our optimization strategy finds a joint update of all the cameras that gives significant improvement of the cost function and therefore can get out of such configurations.

Different comparison methods: Before we proceed any further, we wish to draw attention to an important issue with regard to the method of determining the quality of a result. To analyze the accuracy of a result, we need to first align it to the ground truth rotations in a common frame of reference. While this alignment is done using the averaging of absolute rotations we can choose different loss functions to determine the cost function to be minimized. In our experiments we have used the sum of squares loss functions (analogous to ℓ_2) for the purpose of alignment. Alternatively, one may also use the sum of absolute errors (analogous to ℓ_1), which can be seen to be favorable for the median error metric that is often reported in the literature. However, we believe that the sum of absolute errors can result in misleading interpretations of the quality of relative rotation averaging results. Consider the case where in a result a certain fraction of the estimated rotations are grossly erroneous. The approach of aligning with respect to the ground truth using the sum of absolute errors will ignore these rotations and result in a low estimate for the median rotation error. This low error is of course misleading since some of the rotations are grossly wrong. In contrast, our choice of using the sum of squared errors as the loss function for alignment will faithfully report the overall quality of the estimate, even when the measure being reported is the median error. In the interest of completeness, in Table 6 we show a comparison between the measured median errors for both the sum of squared and sum of absolute errors based alignment for our $\ell_{\frac{1}{2}}$ optimization across all the datasets. As can be seen, the median error reported is always favorable when we use the sum of absolute errors for aligning the estimated rotations with the ground truth. For these reasons, in the results reported in Table 5, we use the sum of squares error for alignment.

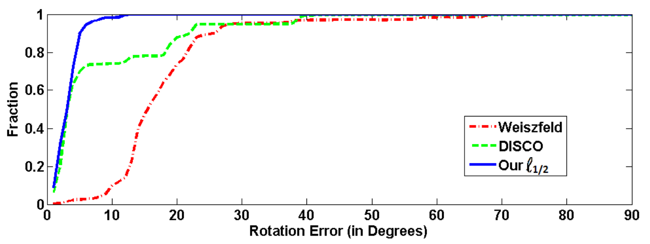
Speed: In Table 5, we also have tabulated the time taken by different methods. Our method is found to be consistently faster than other state-of-the-art methods. Please note that while our method and the Weiszfeld method are tested using a single thread MATLAB implementation on a 2.67GHz desktop, the method of DISCO is tested on a 36 node cluster where each node consists of 2 2.67GHz quad-core processors. Therefore, on a true basis of comparison of computational cost, the method of DISCO is the most expensive method by a significant degree.

Error Distribution: In addition to the comparison in Table 5, in Figure 5, we present a more detailed interpretation of the relative performance of different methods on the 'San Francisco' (SNF) data set which is the largest data set we have used. In Figure 5(a), we compare the distribution of errors for all methods and the leftward shift of our error distribution implies that our rotation estimates are significantly superior to that of DISCO and the Weiszfeld method. Another view of the same comparison can be obtained by considering the cumulative distributions of the individual methods as shown in Figure 5(b). Our cumulative distribution curve can be seen to be significantly higher (i.e. better) compared to the other two methods.

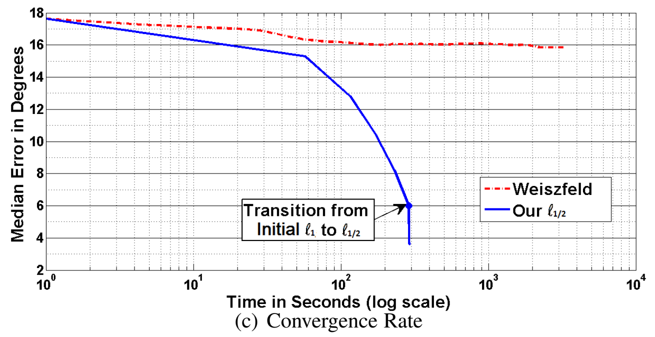
Convergence Rate: In Figure 5(c), we show the convergence behavior of the Weiszfeld method of [13] and our method by plotting the median error as a function of computational time. Note that the scale for the iterations is logarithmic as the Weiszfeld method takes 3186 seconds to converge while our



(a) Error Histograms



(b) Precision-Recall



(c) Convergence Rate

Fig. 5. Comparison of our methods with DISCO [11] and Weiszfeld method [13] on the ‘San Francisco’ (SNF) data set. (a) shows the histograms of errors of individual camera rotation estimates for different methods; (b) represents the fraction of errors below a given error threshold; (c) plots the median error vs. computational time for the Weiszfeld method and our method. Note that the scale for time (x-axis) is logarithmic.

method converges significantly faster. The comparative behaviour of the convergence curves is demonstrable evidence that the Weiszfeld update is unsuitable for large graphs and we should use a joint update of all rotations as is done by our method.

Finally, a comparison of our relative rotation averaging method with a Levenberg-Marquardt based method implemented using Ceres solver (http://ceres-solver.org/) shows the superiority of our method in terms of the accuracy and efficiency but the comparison is beyond the scope and page-limit of the paper.

7 Conclusion

We have developed an efficient method for robust averaging of relative rotations. Our approach admits the use of different robust loss functions. In algorithmic terms, our approach carries out a joint estimation using IRLS that works in the Lie algebra of the rotation group $SO(3)$. Our method significantly outperforms existing approaches in the literature both in terms of speed and accuracy as demonstrated on a large number of real-world data sets.

Appendix

First order expansion of $\mathbf{r}_{ij}$

In this Appendix, we derive the first order expansion of $\mathbf{r}_{ij}$ for Equation 17. Consider two rotations $\mathbf{u}$ and $\mathbf{v}$ in their axis-angle forms s.t. $\theta = \|\mathbf{u}\|$ and $\phi = \|\mathbf{v}\|$. Their quaternion forms are $\left[\cos\left(\frac{\theta}{2}\right), \frac{g(\theta)}{2}\mathbf{u}\right]$ and $\left[\cos\left(\frac{\phi}{2}\right), \frac{g(\phi)}{2}\mathbf{v}\right]$ respectively, where $g(x) = \begin{cases} \frac{\sin(x/2)}{x/2} & \text{if } x \neq 0 \\ 1 & \text{if } x = 0 \end{cases}$.
Note that, $g'(0) = 0$. The composition of $\mathbf{u}$ operating after $\mathbf{v}$ can be written using the quaternion multiplication rule as $\left[\left(\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) - \frac{g(\theta)g(\phi)}{4}\mathbf{u}^T\mathbf{v}\right), \left(a\frac{\mathbf{u}}{2} + b\frac{\mathbf{v}}{2} + c\frac{\mathbf{u}\times\mathbf{v}}{4}\right)\right]$ where $a = \cos\left(\frac{\phi}{2}\right) g(\theta)$; $b = \cos\left(\frac{\theta}{2}\right) g(\phi)$ and $c = g(\theta)g(\phi)$. Let us denote the axis-angle representation of this composition of $\mathbf{u}$ operating after $\mathbf{v}$ as $\mathbf{w}$, i.e. $\mathbf{w} = \mathbf{u} \circ \mathbf{v}$. Let $\psi = \|\mathbf{w}\|$. Therefore, $\sin\left(\frac{\psi}{2}\right)\frac{\mathbf{w}}{\psi} = a\frac{\mathbf{u}}{2} + b\frac{\mathbf{v}}{2} + c\frac{\mathbf{u}\times\mathbf{v}}{4}$. Now, for $\theta < \pi$ and sufficiently small ϕ, $\left(\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) - \frac{g(\theta)g(\phi)}{4}\mathbf{u}^T\mathbf{v}\right)$ is non-negative and then we can write

$$\mathbf{w} = 2\frac{\sin^{-1}(d)}{d}\left(a\frac{\mathbf{u}}{2} + b\frac{\mathbf{v}}{2} + c\frac{\mathbf{u}\times\mathbf{v}}{4}\right) \tag{25}$$

where $d = \left\|a\frac{\mathbf{u}}{2} + b\frac{\mathbf{v}}{2} + c\frac{\mathbf{u}\times\mathbf{v}}{4}\right\|$
When we want to find the Jacobian of $\mathbf{w}$ with respect to $\mathbf{v}$ at the point $\mathbf{v} = \mathbf{0}$ for a given $\mathbf{u}$, the argument or independent variable is $\mathbf{v}$, and $\mathbf{u}$ is treated as a parameter. This Jacobian is given by $\mathbb{J}(\mathbf{w}(\mathbf{0}))$=$m\mathbf{n} + \mathbf{p}\mathbf{q}^T$, where

$$\begin{aligned}
m &= \left[2\frac{\sin^{-1} d}{d}\right]_{\mathbf{v}=\mathbf{0}} \\
&= \frac{2}{g(\theta)} && (26)\\
\mathbf{n} &= \left[\mathbb{J}\left(a\frac{\mathbf{u}}{2} + b\frac{\mathbf{v}}{2} + c\frac{\mathbf{u}\times\mathbf{v}}{4}\right)\right]_{\mathbf{v}=\mathbf{0}} \\
&= \frac{1}{2}\cos\left(\frac{\theta}{2}\right) I_3 + \frac{1}{4}g(\theta)[\mathbf{u}]_\times && (27)\\
\mathbf{p} &= \left[a\frac{\mathbf{u}}{2} + b\frac{\mathbf{v}}{2} + c\frac{\mathbf{u}\times\mathbf{v}}{4}\right]_{\mathbf{v}=\mathbf{0}} && (28)\\
&= \frac{g(\theta)}{2}\mathbf{u} && (29)\\
\mathbf{q} &= \left[\nabla\left(2\frac{\sin^{-1} d}{d}\right)\right]_{\mathbf{v}=\mathbf{0}} \\
&= 2\left[\frac{\frac{d}{\sqrt{1-d^2}} - \sin^{-1} d}{d^2}\nabla d\right]_{\mathbf{v}=\mathbf{0}} \\
&= 2\left[\left(\frac{\frac{d}{\sqrt{1-d^2}} - \sin^{-1} d}{d^3}\right)\mathbb{J}\left(a\frac{\mathbf{u}}{2} + b\frac{\mathbf{v}}{2} + c\frac{\mathbf{u}\times\mathbf{v}}{4}\right)\right. \\
&\qquad \left.\left(a\frac{\mathbf{u}}{2} + b\frac{\mathbf{v}}{2} + c\frac{\mathbf{u}\times\mathbf{v}}{4}\right)\right]_{\mathbf{v}=\mathbf{0}} \\
&= \frac{g(\theta) - \cos\left(\frac{\theta}{2}\right)}{2\sin^2\left(\frac{\theta}{2}\right)}\mathbf{u} && (30)
\end{aligned}$$

Therefore,

$$\mathbb{J}(\mathbf{w}(\mathbf{0})) = \mathbf{mn} + \mathbf{pq^T} = \alpha\mathbf{I_3} + (\mathbf{1} - \alpha)\frac{\mathbf{uu^T}}{\theta^\mathbf{2}} + \frac{\mathbf{1}}{\mathbf{2}}[\mathbf{u}]_\times \tag{31}$$

where

$$\alpha = \frac{\cos\left(\frac{\theta}{2}\right)}{g(\theta)} \tag{32}$$

Equation 31 holds true for $\theta \neq 0$. Further, it is trivial to show that Equation 31 holds true for $\theta = 0$ if we set $\frac{\mathbf{u}}{\theta} = 0$. Similarly, if $\mathbf{w}$ is the composition of $\mathbf{v}$ operating after $\mathbf{u}$, i.e. $\mathbf{w} = \mathbf{v} \circ \mathbf{u}$, then Jacobian of $\mathbf{w}$ with respect to $\mathbf{v}$ at $\mathbf{v} = \mathbf{0}$ can be written as

$$\mathbb{J}\left(\mathbf{w}\left(\mathbf{0}\right)\right) = \alpha \mathbf{I}_3 + (1-\alpha)\frac{\mathbf{u}\mathbf{u}^T}{\theta^2} - \frac{1}{2}\left[\mathbf{u}\right]_\times \tag{33}$$

Therefore, the first order expansion of $\mathbf{r}_{ij}\left(\Delta\mathbf{\Omega}_\mathcal{V}\right) = \boldsymbol{\omega}\left(\mathbf{R}\left(-\Delta\boldsymbol{\omega}_j\right)\mathbf{R}\left(\Delta\boldsymbol{\omega}_{ij}\right)\mathbf{R}\left(\Delta\boldsymbol{\omega}_i\right)\right)$ can be written as

$$\begin{aligned} & \boldsymbol{\omega}\left(\mathbf{R}\left(-\Delta\boldsymbol{\omega}_j\right)\mathbf{R}\left(\Delta\boldsymbol{\omega}_{ij}\right)\mathbf{R}\left(\Delta\boldsymbol{\omega}_i\right)\right) \\ = \quad & \alpha_{ij}(\Delta\boldsymbol{\omega}_i - \Delta\boldsymbol{\omega}_j) + (1-\alpha_{ij})\frac{\Delta\boldsymbol{\omega}_{ij}\Delta\boldsymbol{\omega}_{ij}^T}{\theta_{ij}^2}(\Delta\boldsymbol{\omega}_i - \Delta\boldsymbol{\omega}_j) \\ & + \frac{1}{2}[\Delta\boldsymbol{\omega}_{ij}]_\times(\Delta\boldsymbol{\omega}_i + \Delta\boldsymbol{\omega}_j) \end{aligned} \tag{34}$$

where $\theta_{ij} = \|\Delta\boldsymbol{\omega}_{ij}\|$ and $\alpha_{ij} = \frac{\cos\left(\frac{\theta_{ij}}{2}\right)}{g(\theta_{ij})}$.

Acknowledgments

The authors thank the Department of Computer Science and Automation, Indian Institute of Science, Bengaluru for the use of their FIST cluster facility for running the code for DISCO [11]. The authors also thank the reviewers of the paper for their thoughtful suggestions.

References

[1] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment – a modern synthesis," in Vision Algorithms: Theory and Practice, 2000.
[2] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," International Journal of Computer Vision, vol. 80, no. 2, pp. 189–210, 2008.
[3] ——, "Skeletal graphs for efficient structure from motion," in CVPR, vol. 1, 2008, p. 2.
[4] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, "Building rome in a day," Communications of the ACM, vol. 54, no. 10, pp. 105–112, 2011.
[5] C. Wu, "Towards linear-time incremental structure from motion," in International Conference on 3D Vision(3DV). IEEE, 2013, pp. 127–134.
[6] V. M. Govindu, "Lie-algebraic averaging for globally consistent motion estimation," in Computer Vision and Pattern Recognition (CVPR), vol. 1. IEEE, 2004, pp. I–684.
[7] ——, Riemannian Computing in Computer Vision. Springer (to appear), 2015, ch. Motion Averaging in 3D Reconstruction Problems.
[8] ——, "Combining two-view constraints for motion estimation," in Computer Vision and Pattern Recognition (CVPR), vol. 2. IEEE, 2001, pp. II–218.
[9] K. Wilson and N. Snavely, "Robust global translations with 1dsfm," in Computer Vision–ECCV 2014. Springer, 2014, pp. 61–75.
[10] F. Lu and Z. Chen, "Newton-type iterative solver for multiple view $l2$ triangulation," arXiv preprint arXiv:1405.3352, 2014.
[11] D. J. Crandall, A. Owens, N. Snavely, and D. Huttenlocher, "Discrete-continuous optimization for large-scale structure from motion," in Computer Vision and Pattern Recognition (CVPR). IEEE, 2011, pp. 3001–3008.
[12] A. Chatterjee and V. M. Govindu, "Efficient and robust large-scale rotation averaging," in International Conference on Computer Vision (ICCV). IEEE, 2013, pp. 521–528.
[13] R. Hartley, K. Aftab, and J. Trumpf, "L1 rotation averaging using the weiszfeld algorithm." in Computer Vision and Pattern Recognition (CVPR). IEEE, 2011, pp. 3041–3048.
[14] K. Kanatani, Group Theoretic Methods in Image Understanding. Springer, 1990.
[15] V. S. Varadarajan, Lie Groups, Lie Algebras and Their Representations. Springer-Verlag, 1984.
[16] J. M. Selig, Geometrical Methods in Robotics. Springer-Verlag, 1986.
[17] K. Engo, "On the bch formula in $\mathfrak{so}(3)$," Department of Informatics, University of Bergen, Tech. Rep., 2000.
[18] R. Szeliski, Computer Vision: Algorithms and Applications. Springer-Verlag, 2010.
[19] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, 2nd ed. Cambridge University Press, 2004.
[20] G. Xu and Z. Zhang, Epipolar geometry in stereo, motion and object recognition: a unified approach. Springer Science & Business Media, 1996, vol. 6.
[21] P. J. Huber, Robust Statistics. John Wiley and Sons, New York, 1981.
[22] J. J. William, Introduction to robust and quasi-robust statistical methods. Springer-Verlag, 1983.
[23] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," Commun. Statist. Theory Methods, vol. 9, 1977.
[24] Y. Dai, J. Trumpf, H. Li, N. Barnes, and R. Hartley, "Rotation averaging with application to camera-rig calibration," in Computer Vision–ACCV 2009. Springer, 2010, pp. 335–346.
[25] R. Hartley, J. Trumpf, Y. Dai, and HongdongLi, "Rotation averaging," International journal of computer vision (IJCV), vol. 103, no. 3, pp. 267–305, 2013.
[26] E. Candes and J. Romberg, "L1-magic: recovery of sparse signals via convex programming," http://users.ece.gatech.edu/~justin/l1magic.
[27] V. M. Govindu, "Robustness in motion averaging," in ACCV. Springer, 2006, pp. 457–466.
[28] C. Zach, M. Klopschitz, and M. Pollefeys, "Disambiguating visual relations using loop constraints," in Computer Vision and Pattern Recognition (CVPR). IEEE, 2010, pp. 1426–1433.
[29] R. Tron, R. Vidal, and A. Terzis, "Distributed pose averaging in camera networks via consensus on SE(3)," in Distributed Smart Cameras (ICDSC). IEEE, 2008, pp. 1–10.
[30] R. Tron and R. Vidal, "Distributed computer vision algorithms through distributed averaging," in Computer Vision and Pattern Recognition (CVPR). IEEE, 2011, pp. 57–63.
[31] 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, 2015.
[32] R. Tron, Distributed Optimization on Manifolds for Consensus Algorithms and Camera Network Localization. Johns Hopkins University, 2012.
[33] J. Fredriksson and C. Olsson, "Simultaneous multiple rotation averaging using lagrangian duality," in Computer Vision–ACCV 2012. Springer, 2013, pp. 245–258.
[34] F. Arrigoni, L. Magri, B. Rossi, P. Fragneto, and A. Fusiello, "Robust absolute rotation estimation via low-rank and sparse matrix decomposition," in International Conference on 3D Vision(3DV). IEEE, 2014.
[35] D. J. Crandall, A. Owens, N. Snavely, and D. P. Huttenlocher, "Sfm with mrfs: Discrete-continuous optimization for large-scale structure from motion," IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 35, no. 12, pp. 2841–2853, 2013.

Avishek Chatterjee received the B.E.E. degree in Electrical Engineering from Jadavpur University, Kolkata, India in 2009 and an M.E. in System Science and Automation from Indian Institute of Science, Bengaluru, India in 2011. He is currently pursuing a Ph.D. in System Science and Signal Processing at the Indian Institute of Science. His current research interests include geometric estimation in computer vision.

Venu Madhav Govindu received the Ph.D. degree in Electrical Engineering from the University of Maryland, College Park, MD, USA. He is currently an Associate Professor with the Department of Electrical Engineering, Indian Institute of Science, Bengaluru, India. His research interests include geometric and statistical inference problems in computer vision.