DISTRIBUTED POSE AVERAGING IN CAMERA SENSOR NETWORKS USING CONSENSUS ON MANIFOLDS

Roberto Tron, René Vidal

Johns Hopkins University
Center for Imaging Science
302B Clark Hall, 3400 N. Charles St.,
Baltimore MD 21218, USA

Andreas Terzis

Johns Hopkins University
Computer Science Department
417 Wyman Park Bldg., 3100 Wyman Park Dr.,
Baltimore MD 21218, USA

ABSTRACT

We have reached a technology inflection point where the combined availability of miniature, network-enabled devices (i.e. motes) and low-power CMOS cameras, provide the opportunity of using camera sensor networks to carry out automatic visual analysis of complex scenes. However, the development of a distributed sensing framework for such networks faces several critical challenges. On the one hand, most existing computer vision algorithms are centralized and require an amount of computation that would easily overwhelm the nodes' limited resources. On the other hand, most existing distributed sensing algorithms such as consensus are designed for low-dimensional and Euclidean data. In contrast, the data that a camera network uses to infer a "model of the observed scene" are typically high-dimensional, and the parameters of these models (e.g., pose) often live in non Euclidean spaces.

In this paper, we propose distributed algorithms for estimating the average pose of an object viewed by all the cameras motes in a network. To this effect, we propose distributed averaging consensus algorithms on the group of 3-D rigid body transformations $SE(3)$. We rigorously analyze the convergence of the proposed algorithms. We also provide synthetic experiments that confirm our analysis and validate our approach.

Index Terms— camera sensor networks; pose estimation; consensus; optimization on manifolds.

1. INTRODUCTION

Recent hardware innovations have produced low-power embedded computers (i.e., motes), equipped with small cameras that can self-organize into wireless mesh networks. These disruptive technologies provide the opportunity to devise compelling new applications at the intersection of sensor networks and computer vision. In particular, one can rethink many of the classical computer vision applications such as structure from motion (SfM), target tracking, object recognition, etc., in the context of scenes observed by a large number of cameras.

A number of fundamental challenges must be addressed for this promise to be fulfilled. Specifically, most computer vision algorithms assume that all the data (i.e., the images) are available on a single computer where centralized processing is possible. This paradigm however, is inherently incompatible with sensor networks, because it requires the transmission of large volumes of raw data and processing resources not available in mote-class devices. One is thereby naturally drawn to distributed algorithms, in which motes analyze the raw data locally and collaborate to reach a shared, global analysis of the scene. This approach is intuitively attractive, because it affords nodes to exchange only distilled information that is relevant to the collaboration. Because such information can be expressed in compact form for most applications –the pose of an object, for example, lives in a six-dimensional manifold– this approach can result in radical energy savings.

Prior work. Unfortunately, existing distributed sensing algorithms cannot fulfill this goal. For example, *distributed consensus algorithms*[1] that have been used extensively in sensor networks (see [**?**, **?**, **?**, **?**, **?**] and the references therein) operate on low-dimensional measurements (e.g., temperature) which lie in a Euclidean space. On the other hand, most of the quantities involved in the visual representation of a scene are not directly measurable (i.e., they can not be directly extracted from what nodes see) and belong to spaces with rich and complex non-Euclidean structures (e.g., Lie groups). Consequently, a principled treatment of such data requires the use of optimization on manifolds. While optimization on manifolds has been widely used in computer vision problems such as 3D reconstruction [**?**], 3D motion segmentation [**?**], and manifold clustering [**?**], relatively less work has been done in extending consensus algorithms to non-Euclidean data [**?**].

There are a few papers addressing localization of camera networks [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**]. Semi-automatic camera network calibration methods use laser printed textures mounted on a board [**?**] or modulated LED emissions [**?**, **?**]. Automatic methods perform local SfM via robust bundle adjustment [**?**],

[1] Other names have also been used such as information consensus, consensus averaging, agreement protocols, etc.

and integrate the information using belief propagation [**?**]. We posit that a natural framework for obtaining a consistent estimate of the pose of each camera is by using consensus. The key question is how to design a local SfM algorithm that would (1) replace the local average algorithm in the classical consensus problem and (2) when iteratively applied, converge to a consistent global estimate of all 3D poses. To the best of our knowledge, there is no prior work on fully automatic and distributed camera sensor network localization using extensions of consensus algorithms. Also, there are no transcriptions of SfM constraints into distributed averaging.

Paper contributions. This paper makes the following three contributions. First, we demonstrate that generalizing existing consensus algorithms from the Euclidean to the non Euclidean setting is not trivial. We discuss the related obstacles and outline the nature of the desired solutions through our algorithms. Second, we present distributed algorithms for solving a simple yet foundational computer vision problem: estimating the pose of an object viewed by all the cameras in a network. More specifically, we devise distributed algorithms for performing consensus on the group of 3-D rigid motions in order to average the pose of the object in question. Our algorithms make use of the geodesic distance in $SE(3)$, in contrast to the Euclidean distance adopted by traditional consensus approaches. As a result, the average pose is estimated as the Karcher mean on this manifold. Our contribution is precisely to propose algorithms for estimating the Karcher mean in a distributed fashion. These algorithms lay the foundation necessary for future directions of research in this area. Finally, we analytically prove the convergence of the proposed algorithms and experimentally evaluate their performance using simulations.

Paper outline. The remainder of the paper is organized as follows. We first present a review of traditional consensus algorithms that deal with Euclidean data. We also review the basics of differential geometry with a focus on optimization on manifolds. Having introduced the necessary mathematical framework, we introduce our generalizations of classical consensus approaches to non-Euclidean data. Since our algorithms are iterative in nature, we also discuss their convergence properties. Finally, we present results on synthetic data that validate the different aspects of our proposed framework.

2. MATHEMATICAL BACKGROUND

In this section, we review some basic concepts related to Euclidean consensus algorithms and differential geometry that are relevant to our algorithms.

2.1. Review of Euclidean consensus algorithms

Consider a sensor network with N nodes. The communication in the sensor network can then be represented using an undirected graph $G = (V, E)$. The set $V = \{v_1, \ldots, v_N\}$ contains the vertices of the graph, where $v_i \in V$ represents the i^{th} camera sensor. The set E contains the edges of the graph and helps describe the communication links between these nodes. More specifically, if nodes i and j can directly communicate with each other, then the pair $\{i, j\} \in E$. Given this construction, we define the *adjacency matrix* $A \in \mathbb{R}^{|V|\times|V|}$ of the graph as

$$[A]_{ij} = \begin{cases} 1 & \text{if } \{i,j\} \in E \\ 0 & \text{otherwise} \end{cases}. \tag{2.1}$$

Note that A is symmetric since G is undirected. In the Euclidean setting, we associate with each node $i = 1, \ldots, N$, a scalar *state* $x_i \in \mathbb{R}$. We then say that the nodes have reached *consensus* when the condition

$$x_1 = x_2 = \ldots = x_N = \alpha \tag{2.2}$$

is satisfied. The value α is called the *collective decision* of the network.

A *consensus algorithm* defines a *protocol*, i.e., a rule that each node has to follow to update its state so that the nodes reach a consensus. To this effect, the task of a *distributed consensus algorithm* is to define a protocol which uses only interactions between neighboring nodes.

A popular (discrete time) protocol for iteratively computing the mean of the initial states $\alpha = \frac{1}{n}\sum_{i=1}^{N} x_i$ is given by

$$x_i^{(k+1)} = x_i^{(k)} + \varepsilon \sum_{j\in N_i} a_{ij}(x_j^{(k)} - x_i^{(k)}) \tag{2.3}$$

where N_i is the set of neighbors of the i-th node, $x_i^{(k)}$ is the state of node i during the k-th step of the algorithm and ε is 1 over the maximum node degree. Note that the sum over N_i can be safely extended to $\{1, \ldots, N\}$ because $a_{ij} = 0$ if i and j are not neighbors. Equation (2.3) can then be rewritten in term of the updates $u_i^{(k)} = \varepsilon \sum_{j\in N_i} a_{ij}(x_j^{(k)} - x_i^{(k)})$, as

$$x_i^{(k+1)} = x_i^{(k)} + u_i^{(k)}. \tag{2.4}$$

It is easy to see that (2.4) is in fact a gradient descent algorithm that minimizes the function

$$\varphi = \frac{1}{2} \sum_{\{i,j\}\in E} a_{ij}(x_i - x_j)^2 \tag{2.5}$$

Also, it is easy to check that the mean of the states is preserved at each iteration, i.e.,

$$\sum_{i=1}^{N} x_i^{(k)} = \sum_{i=1}^{N} x_i^{(k+1)} = \alpha. \tag{2.6}$$

Equivalently, the sum of the updates over all the nodes is zero:

$$\sum_{i=1}^{N} u_i^{(k)} = 0. \tag{2.7}$$

Therefore, if the graph G is connected, the minimum of (2.5) is achieved when all the nodes reach a consensus, i.e., when all the states are equal to the average of the initial states. More information on this protocol can be found in [**?**]. Note that this algorithm is readily extended to the case $x_i \in \mathbb{R}^n$ by applying it to each coordinate separately.

2.2. Review of Riemannian Geometry

Given a smooth manifold $\mathcal{M}$, we define a curve in the manifold as a smooth function $\gamma(t) : \mathbb{R} \to \mathcal{M}$. The *tangent space* of $\mathcal{M}$ at a point $x \in \mathcal{M}$, denoted as $T_x\mathcal{M}$, is then defined as the span of the tangent vectors for all the possible curves γ passing through x. A *Riemannian metric* is a continuous collection of dot products $\langle \cdot | \cdot \rangle_x$. Using this metric, we define the length of a curve between two points $x, y \in \mathcal{M}$ as

$$\mathcal{L}_a^b(\gamma) = \int_a^b \langle \dot{\gamma}(t) | \dot{\gamma}(t) \rangle_{\gamma(t)}^{\frac{1}{2}} dt, \tag{2.8}$$

where $\gamma(a) = x$ and $\gamma(b) = y$. A curve between x and y with minimum length is called a $geodesic$. The distance between two points in the manifold is subsequently defined as the length of the geodesic curve between them

$$d(x, y) = \mathcal{L}_0^1(\gamma),\ \gamma(0) = x, \gamma(1) = y. \tag{2.9}$$

We can then define the *exponential map* $\exp_x(v) : T_x\mathcal{M} \to \mathcal{M}$, which maps each tangent vector $v \in T_x\mathcal{M}$ to the point in $\mathcal{M}$ obtained by following the geodesic passing through x with direction $\frac{v}{\|v\|}$ for a distance $\|v\|$. The exponential map is a diffeomorphism between a sufficiently small neighborhood of 0 in $T_x\mathcal{M}$ and a neighborhood of x in $\mathcal{M}$. The *logarithm map* is the inverse of the exponential map and is denoted as $\log_x = \exp_x^{-1}$. Note that if $y = \exp_x(v)$ then

$$\| \log_x(y) \| = \|v\| = d(x, y). \tag{2.10}$$

2.3. Lie groups

A Lie group G is an algebraic group which can also be considered as a differentiable Riemannian manifold. In particular, the group is characterized by a unique identity element e and two group operations

$$\begin{aligned} \textit{multiplication} \quad & g_1 g_2 : G \times G \to G, \text{ and} \\ \textit{inversion} \quad & g^{-1} : G \to G, \end{aligned}$$

which are differentiable mappings. The tangent space at the identity is called the Lie algebra of the group. We denote as $\exp(\cdot)$ and $\log(\cdot)$ the exponential map and logarithmic map at the identity e. These mappings at a generic point $X \in G$ can be computed using *parallel transport* as:

$$\exp_X(A) = X \exp(X^{-1}A), \tag{2.11}$$

$$\log_X(B) = X \log(X^{-1}B), \tag{2.12}$$

with $A \in T_X G$ and $B \in G$.

In this paper we are particularly interested in the special orthogonal group $SO(3)$. This is the group of orthogonal 3×3 matrices whose determinant is equal to one, i.e., the group of rotation matrices. The Lie algebra for this group is $so(3)$, the space of 3×3 skew-symmetric matrices. In this case the exponential map at the identity $\exp(w)$, $w \in so(3)$, can be defined using the well-known Rodrigues' formula [**?**], as

$$\exp(w) = I + \frac{\hat{w}}{\|w\|} \sin(\|w\|) + \frac{\hat{w}^2}{\|w\|^2}[1 - \cos(\|w\|)]. \tag{2.13}$$

One can then derive the logarithm map at the identity as

$$\theta(R) = \arccos\left(\frac{\operatorname{tr}(R) - 1}{2}\right), \tag{2.14}$$

$$\log(R) = \begin{cases} \dfrac{1}{2\sin\theta}(R - R^\top) & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0. \end{cases} \tag{2.15}$$

where $\operatorname{tr}(R)$ is the trace of the matrix. It can be shown that the geodesic distance in $SO(3)$ is given by

$$d^2(R, Q) = -\frac{1}{2}\operatorname{tr}\{[\log(R^\top Q)]^2\} \quad R, Q \in SO(3). \tag{2.16}$$

Given a function $f : SO(3) \to \mathbb{R}$, the *covariant derivative* of f at $R \in SO(3)$ (denoted as $\nabla_x f \in so(3)$) is the unique vector defined as

$$\operatorname{tr}(v^\top \nabla f) = \left. \frac{d}{dt} f(\gamma(t)) \right|_{t=0}, \tag{2.17}$$

where $v \in T_R SO(3)$ and $\gamma(t)$ is the geodesic passing through R in the direction of v.

Note that the covariant derivative of a function on the manifold can be treated as the equivalent of the conventional derivative of a function defined on a Euclidean manifold. Therefore, gradient descent like approaches would use the covariant derivative of the squared distance with respect to one of the arguments. This covariant derivative can be explicitly evaluated (see [**?**] for a proof) as

$$\nabla_R d^2(R, Q) = -R \log(R^\top Q). \tag{2.18}$$

2.4. Karcher mean

Let $x_i, i = 1, \ldots, N$ be a set of points in a smooth manifold $\mathcal{M}$, for which a Riemannian metric can be defined. The metric can then be used to obtain the geometric distance $d(x, y)$ between any two points $x, y \in \mathcal{M}$. The Karcher mean of this set is defined as the point $\bar{x} \in \mathcal{M}$ for which the sum of squared distances

$$\sum_{i=1}^{N} d^2(x_i, \bar{x}) \tag{2.19}$$

is minimized. It can be shown ([**?**]) that a necessary and sufficient condition for $\bar{x}$ to be the Karcher mean is

$$\sum_{i=1}^{N} \log(x_i^{-1}\bar{x}) = 0 \tag{2.20}$$

where $\log$ is the logarithmic map at the identity. This condition helps us describe an iterative algorithm to compute the Karcher mean ([**?**, **?**]), the details of which are as follows.

1. Set initial mean in $\mathcal{M}$ as $\bar{x} = x_1$.
2. Compute the mean in $T_{\bar{x}}\mathcal{M}$ as $w = \frac{1}{n}\sum_{i=1}^{n} \log_{\bar{x}}(x_i)$.
3. While $\|w\| < \delta$, update $\bar{x}$ to $\bar{x} \leftarrow \exp_{\bar{x}}(\varepsilon w)$, for $\varepsilon \leq 1$ and go to step 2.

3. DISTRIBUTED POSE ESTIMATION VIA CONSENSUS ON MANIFOLDS

In what follows, we consider an object in a scene that is assumed to be visible to all the N cameras in the network. Furthermore, we assume that each of these $i = 1, \ldots, N$ nodes can estimate the position of the object, as a pair of rotation and translation g_i lying in $SO(3) \times \mathbb{R}^3$ (direct product of the special orthogonal group and the three-dimensional Euclidean space). Our goal then is to estimate an average pose $\bar{g}$ from all the measurements in a distributed fashion (i.e. by exchanging data only between neighboring pairs of nodes).

In order to propose a consensus algorithm for estimating the pose $\bar{g}$, we essentially need to deal with the special Euclidean group i.e., $SE(3)$. It has been shown in [**?**] that there is no Riemannian metric, for which the geodesics can be obtained from the associated Lie algebra $se(3)$ (i.e., screw motions). To solve this issue, we notice that $SE(3) = SO(3) \times \mathbb{R}^3$. In such a scenario, one can use a standard result that when the data lie on a manifold which is the direct product of two spaces, one can separately consider the metrics and covariant derivatives associated with the two spaces [**?**]. Therefore, we perform our optimization on the individual spaces $SO(3)$ and $\mathbb{R}^3$. Note that in the case of $\mathbb{R}^3$, we can employ traditional Euclidean consensus algorithms as discussed in Section 3. Hence, we focus our attention on consensus algorithms for $SO(3)$, i.e., the rotation component of the pose of the object.

3.1. Consensus in the Manifold

Our first algorithm, which we refer to as *consensus in the manifold*, can be considered as a direct extension of the classic consensus algorithm to the non-Euclidean case. Essentially, we follow a Riemannian gradient decent scheme for minimizing our cost function, as opposed to the traditional gradient decent method. In particular, we substitute the derivatives of the related objective function in the Euclidean space, with covariant derivatives on the manifold. Consequently, the update step is modified as a move along the geodesics defined by the covariant derivatives of the objective function. In this manner, we ensure that the solution estimated by the optimization scheme always lies on the required manifold.

Let us denote the measurement of the object's pose from the i-th node as the pair $g_i = (R_i, p_i)$, where $R_i \in SO(3)$ and $p_i \in \mathbb{R}^3$. We then define the potential function φ that is to be minimized in order to estimate the mean rotation, as

$$\varphi(R_1, \ldots, R_n) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} d^2(R_i, R_j), \tag{3.1}$$

where $d(\cdot,\cdot)$ is the distance between two elements of $SO(3)$. The covariant derivative of φ with respect to the k-th rotation, can then be explicitly calculated as

$$\nabla_{R_k}\varphi = \nabla_{R_k}\sum_{i\neq k} a_{ik} d^2(R_i, R_k) = -\sum_{i=1}^{n} a_{ik} R_k \log(R_k^\top R_i). \tag{3.2}$$

In the simplification of the above equation, we used the facts that $a_{ij} = a_{ji}$, $d^2(\cdot,\cdot)$ is symmetric and $d^2(R_k, R_k) = 0$.

Our proposed consensus protocol on $SO(3)$ corresponds to using a Riemannian gradient descent search for minimizing the cost function φ. In what follows, we use $R_k^{(l)}$ to denote the estimate of R_k at the l^{th} iteration of our algorithm. Essentially, $R_k^{(l)}$ is updated along the geodesic corresponding to the covariant derivative direction $-\nabla_{R_k^{(l)}}\varphi$ with a step size ε, as

$$\begin{aligned} R_k^{(l+1)} &= \exp_{R_k^{(l)}}(-\varepsilon \nabla_{R_k^{(l)}}\varphi) \\ &= R_k^{(l)} \exp\left({R_k^{(l)}}^\top R_k^{(l)} \varepsilon \sum_{i=1}^{n} a_{ik} \log({R_k^{(l)}}^\top R_i^{(l)}\right) \\ &= R_k^{(l)} \exp\left(\varepsilon \sum_{i=1}^{n} a_{ik} \log\left({R_k^{(l)}}^\top R_i^{(l)}\right)\right) \end{aligned} \tag{3.3}$$

This update for $R_k^{(l)}$ can be rewritten more compactly as

$$R_k^{(l+1)} = R_k^{(l)} U_k^{(l)}, \tag{3.4}$$

where the update U_k is defined as

$$U_k^{(l)} = \exp(\varepsilon \sum_{i=1}^{n} a_{ik} \log({R_k^{(l)}}^\top R_i^{(l)}). \tag{3.5}$$

Notice that all the terms in (3.1) are non-negative, i.e., $a_{ij} \geq 0$ and $d^2(R_i, R_j) \geq 0$. Therefore, we conclude that the cost function φ is non-negative for all $R_1, \ldots, R_n \in SO(3)$. In addition, we note that $\varphi = 0$ implies that for each pair $\{i, j\} \in E$, we have either $a_{ij} = 0$ or $d^2(R_i, R_j) = 0$. Also notice that the distance $d^2(R_i, R_j) = 0$ if and only if the associated rotations are equal, i.e., $R_i = R_j$. Hence, if we assume that G is strongly connected, we can conclude from the above that φ achieves its global minimum $\varphi = 0$, if and only if

$\forall i, j : R_i = R_j$. It is important to note that the minimizer of φ is not unique. This can be verified by observing that $\varphi = 0$ whenever $\forall i = 1, \ldots, N, R_i = R_c \in SO(3)$. The non-uniqueness of R_c prevents the existence of a unique minimizer.

Since the cost function φ is non-negative, we note that it is bounded from below. Therefore, our proposed consensus protocol is guaranteed to converge to a solution, since it descends the potential function φ at each iteration of the algorithm. We note that due to its design, our algorithm can potentially converge to any one of the possible minimizers. However, in our experiments, we have noticed that this protocol always converges to the desired global minimizer of φ. Our ongoing research aims at investigating whether the proposed protocol can converge to any of the undesired potential minimizers, and if so, under what conditions does this happen?

We now analyze the nature of the solution estimated by our algorithm. If we denote the Karcher mean of the rotations R_k at the l-th iteration by $\bar{R}$, we see that we have the relation

$$\sum_{i=1}^{n} \log(\bar{R}^\top R_k^{(l)}) = 0. \tag{3.6}$$

Now, recall the Campbell-Baker-Hausdorff (CBH) formula:

$$\exp(A)\exp(B) = \exp(C), \tag{3.7}$$

where A,B,C are elements of the Lie algebra $so(3)$, and

$$\begin{aligned} C &= A + B + \frac{1}{2}[A,B] + \frac{1}{12}\big([A,[A,B]] + [B,[B,A]]\big) + \ldots] \\ &= A + B + \mathcal{O}(|(A,B)|) \end{aligned} \tag{3.8}$$

From our consensus protocol, we see that after one iteration:

$$\begin{aligned} &\sum_{i=1}^{n} \log(\bar{R}^\top R_k^{(l+1)}) = \sum_{i=1}^{n} \log(\bar{R}^\top R_k(l) U_k(l)) \\ &= \sum_{i=1}^{n} \log(\bar{R}^\top R_k(l)) + \sum_{i=1}^{n} \log(U_k(l)) \\ &+ \mathcal{O}(|(\bar{R}^\top R_k(l), \log(U_k(l)|) \\ &= 0 + \mathcal{O}(|(\bar{R}^\top R_k(l), \log(U_k(l)|) \end{aligned} \tag{3.9}$$

where we used (3.6) for the last equality. Hence, we see that the Karcher mean at iteration l is equal to the Karcher mean at iteration $l+1$, if the higher order terms can be ignored.

It is interesting to consider the special case where all the rotations have a common axis and belong to a one-parameter family in $SO(3)$. This happens, for instance, when all the rotation are inside a single plane in $\mathbb{R}^3$. In this particular situation, it is possible to write $R_i = \exp(t_i w)$ where $w \in so(3)$ represents the common rotation axis and $t_i \in \mathbb{R}$ are the rotation angles. As a consequence, all the rotation updates $U_k^{(l)}$ also belong to the same family. Hence, they commute with each of the rotations, i.e. $R_i^{(l)} U_j^{(l)} = U_j^{(l)} R_i^{(l)}$, $i, j \in 1, \ldots, N$. When this happens, it implies that the Lie brackets and the other higher order terms of (3.9) are zero and the algorithm therefore converges to the correct mean.

3.2. Consensus in the tangent space

As we have seen above, the Karcher mean of the data estimated using the earlier proposed algorithm does not necessarily remain constant at each iteration. In order to overcome this issue, we propose a second method that is derived from the global iterative algorithm used for estimating the Karcher mean. In this algorithm, we work on the tangent space to the manifold to perform the consensus as opposed to working on the manifold itself. We refer to this algorithm as *consensus in the tangent space*. This method has the same convergence properties as the global approach, but requires a common initialization for all the nodes. In addition to our existing notation, we introduce the term R_{cost} to represent the predefined initialization for the rotations. In this consensus algorithm, all the nodes start with a common initialization $R_i^{(1)} = R_{cost}$. At each iteration l, every node i computes the covariant derivative of the distance between the common estimate $R_i^{(l)}$ and its own rotation R_i as

$$\nabla_{R_i^{(l)}} d^2(R_i^{(l)}, R_i) = R_i^{(l)} \log({R_i^{(l)}}^\top R_i) \tag{3.10}$$

The rotation at each node is then updated with a rotation U that is computed as

$$U = \exp\left(\frac{1}{n}\sum_{i=1}^{N} \log\left({R_i^{(l)}}^\top R_i\right)\right) \tag{3.11}$$

Since the result of the logarithmic map is homeomorphic to $\mathbb{R}^3$, the quantity $\frac{1}{n}\sum_{i=1}^{N} \log({R_i^{(l)}}^\top R_i)$ can be computed using an Euclidean consensus algorithm. Upon convergence, all the nodes share the same value of U and we compute $R_i^{(l+1)}$ as

$$R_i^{(l+1)} = R_i^{(l)} U. \tag{3.12}$$

Note that since the initial rotations $R_i^{(1)}$ were set to the same value, the estimates $R_i^{(l)}$ at the end of the i^{th} step, will be the same for all the nodes. Therefore, we estimate the Karcher mean by using the consensus algorithm in the tangent space, that proceeds as follows.

1. Set initial rotations as $R_i^{(1)} = R_{cost} \quad \forall i = 1, \ldots, N$.
2. Compute the mean $w = \frac{1}{n}\sum_{i=1}^{N} \log({R_i^{(l)}}^\top R_i)$ across the network using an Euclidean consensus algorithm
3. While $\|w\| < \delta$, for some predefined δ, compute the updates as

   $$R_i^{(l+1)} = R_i^{(l)} \exp(w) \quad \forall i = 1, \ldots, N$$

   and repeat step 2.

By comparing this procedure with the global algorithm used for estimating the Karcher mean, it is easy to see that both the

algorithms update the mean through the same sequence of updates. Therefore, the consensus algorithm in the tangent space has the same convergence properties as the global algorithm. However, there is a potential problem with this approach. The initial estimate for the mean is initialized to R_{cost} independently from the actual data R_i. It could so happen that R_{cost} is in the cut locus of one of the R_i. In this case, the logarithmic map is not defined and the algorithm's framework breaks down. Hence, in order to deal with this issue, we propose an algorithm that merges the above proposed algorithms.

3.3. Extension to measurements of object's pose in different frames of reference

Before proceeding to the case of merging the proposed algorithms, we address the issue of selecting the reference frames for measuring the pose. In the discussion above, we have implicitly used the fact that all the rotations $\{R_i\}_{i=1}^n$ are provided with respect to the same reference frame. In this section, we relax this condition and extend our algorithms to the case where each node has its own reference frame. Let us introduce some additional notation for this purpose. Assume an arbitrary origin in the space, given by $h_0 = (S_0, t_0)$, with $S_0 \in SO(3)$ and $t_0 \in \mathbb{R}^3$. For the sake of simplicity, we choose $S_0 = I$, the identity matrix and $h_0 = \mathbf{0}$. We then define $h_i = (S_i, t_i),\ i = 1, \ldots, n$ as the positions of the nodes with respect to this reference. In order to deal with the independent reference frames for each node, we require that the network be localized. The network is said to be *localized* if each node $i = \{1, \ldots, n\}$ knows the location of node j (where $\{i, j\} \in E$) with respect to its frame of reference.

We define the term $h_{ij} = (S_{ij}, t_{ij}) \in SE(3), i = \{1, \ldots, n\}$, $\{i, j\} \in E$, to indicate the position of node i in the reference system of j. Note that this also corresponds to the mapping of the points' locations from the reference frame of i to the reference frame of j. In fact, it can easily be verified that

$$h_{ij} = h_j h_i^{-1} \tag{3.13}$$

This change of coordinates gives us a way to move from one frame of reference to the other. If R_i and t_i are the rotation and translation expressed in the reference frame of node i, these quantities can be expressed in the reference frame of node j as

$$(R_j, t_j) = (S_{ij} R_i, S_{ij} t_i + t_{ij}) \tag{3.14}$$

This provides us with a solution to relax the constraint that all the rotations need to be expressed in the same co-ordinate system.

3.3.1. Consensus in the manifold

We first note that the definition of distance in (2.9) does not depend on the choice of S_0. If we apply any arbitrary rotation $R_0 \in SO(3)$ to both arguments, the value of function does not change i.e., $d^2(R, R') = d^2(R_0 R, R_0 R')$, $R, R', R_0 \in SO(3)$. Similarly, we also see that the potential function φ in (3.1) and the update matrices defined in (3.5) for the consensus in the manifold protocol, do not depend upon the choice of S_0. In particular, this means that, even if each node "views" the data $\{R_i\}_{i=1,\ldots,n}$ with respect to a different reference frame, the estimation of the update matrices $U_k^{(l)}$ would be accurate as long as the data at each node is expressed with respect to the same reference frame. More explicitly, let $\{R_i\}_{i=1,\ldots,n}$ be the data expressed using S_0 as the reference frame. Also, define $\{R_{ij}\}_{i=1,\ldots,n}$ to express the same data using the reference frame of the j^{th} node. When the reference frame is transferred from that of the j^{th} node to the k^{th} node, we note that the data's representation is transformed as

$$R_{jk} = S_{jk} R_j. \tag{3.15}$$

3.3.2. Consensus in the tangent space

Since $\log({R_i^{(l)}}^\top S^\top S R_i) = \log({R_i^{(l)}}^\top R_i)$ for any $S \in SO(3)$, we argue that the update matrices U for the consensus algorithm in the tangent space can be computed as described, as long as R_i and $R_i^{(l)}$ are expressed in the same coordinate system. This only requires a modification in the manner each node is initialized. In fact, the values R_i^1 have to be consistent with the localization of the network. For instance, if we set $R_1^{(1)} = R_{cost}$, then we have to set $R_j^{(1)} = S_{1j} R_j$, $j = 2, \ldots, N$. If the localization is not known a priori, we would require a method to generate a coherent initialization. Motivated by this fact, we propose our final algorithm in what follows. Note that, although we have limited our analysis to the estimation of rotations, a similar argument can be extended for the case of translations. Thus, our proposed algorithms can be used even when the nodes have different reference frames.

3.4. Combined Algorithm

To summarize, if we perform a consensus in the manifold, we have an algorithm that updates each estimate of the pose at the different nodes and converges to a common value. Unfortunately, due to the implicit curvature of $SO(3)$, this common value is only a first order approximation of the true Karcher mean. On the other hand, if we perform a consensus in the tangent space, we have an algorithm that converges correctly, subject to an initialization that is coherent with the localization of the network. Therefore, we propose to combine the two algorithms to get an accurate solution. First, we run the consensus in the manifold until it converges, thereby obtaining a set of rotations which are consistent with the localization of the network and close to the correct Karcher mean. This result is used to initialize for the consensus in the tangent space, which then produces an accurate estimate of the average of the poses.

4. EXPERIMENTS

We tested our algorithms on synthetic data by simulating a network with $N = 20$ nodes connected with a k-regular graph, $k = 6$. We generate the measurements for the nodes starting from the transformation

$$g = \{I, \begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix}\} \tag{4.1}$$

to which we add noise. In $SO(3)$ we use as noise term rotations around a random axis and with an angle generated from a gaussian distribution with variance of 20° but we reject samples outside the interval $[-90^\circ, 90^\circ]$. This is necessary in order to make sure that the Karcher mean is actually unique ([**?**]). For the noise in the translation space, we simply use gaussian noise with variance 0.35.

We used 70 iterations for the consensus in the manifold while for the consensus in the tangent space we used 4 iterations for the main loop and 140 iterations for the Euclidean consensus.

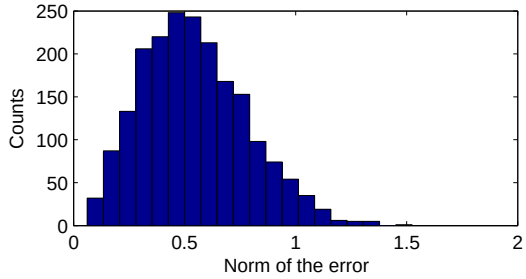
For the combined algorithm, we took the results of the consensus in the manifold and added two iterations of the consensus in the tangent space, again with 140 iteration for the Euclidean consensus.

In our experiments the number of iterations was fixed, but a better implementation would stop when a given tolerance for the convergence is satisfied. Note that the consensus in the tangent space requires very few iterations for the main loop, but it requires a really good convergence of the inner Euclidean consensus (hence the high number of iterations) in order to keep all the nodes aligned with high precision.

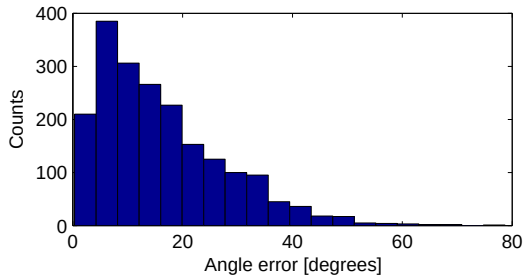
For the algorithms that work with localized networks, the reference frames are randomly generated within a cube of dimension $20 \times 20 \times 20$ and with uniform random orientation. We repeated the experiments 100 times. We have collected the initial errors for the rotations and translations, the final error for the translations (common to all the algorithms) and the final error for the rotations for the three algorithm. Figures 1 and 2 show histograms of these results.

For the rotations we measured the angle of rotation in degrees while for the translation we used the norm of the error. We see that the rotation errors for the consensus in the manifold are in the order of 10^{-2}. As expected they are not zero but, as a first order approximation, they can be considered sufficient low for many applications. The errors for the consensus in the tangent space and the combined algorithm are in the order of 10^{-6} and 10^{-5}, respectively and are actually zero for many of the trials, as one can see from the histograms. In this case, the errors are mostly due to numerical issues.

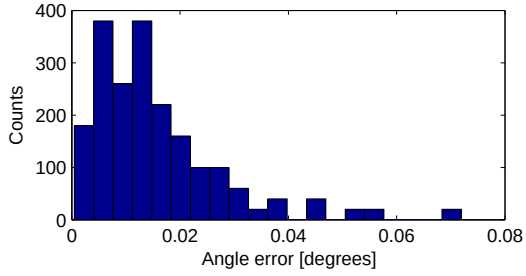
We will now look more in the details of the trial for which the consensus in the manifold gave the worst estimate of the mean. We can plot the angle between the estimate at each node and the mean computed with the global algorithm in function



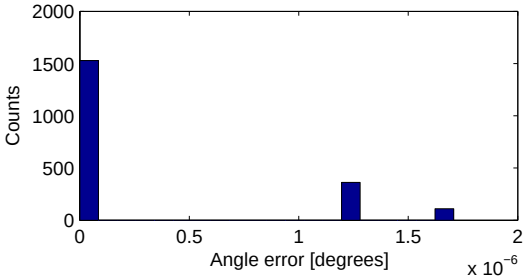
(a) Initial translation errors



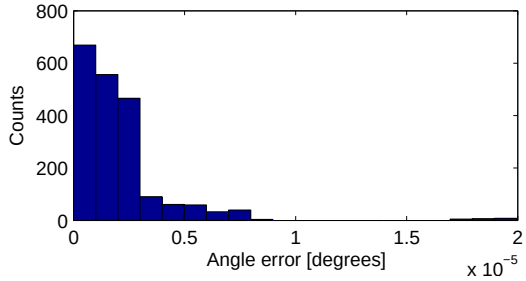
(b) Initial rotation errors



(c) Final rotation errors for the consensus in the manifold

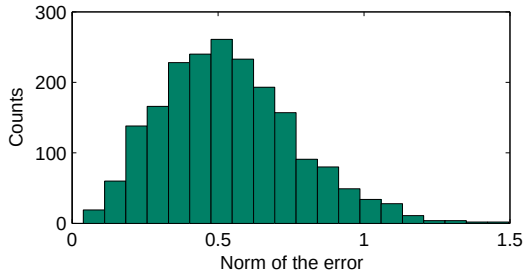


(d) Final rotation errors for the consensus in the tangent space

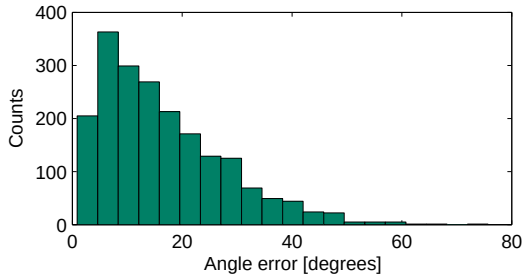


Final rotation errors for the combined algorithm

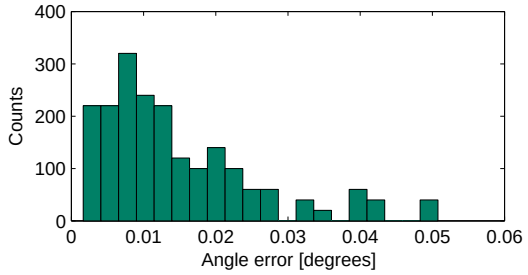
Fig. 1. Simulation results (translation and rotation errors)



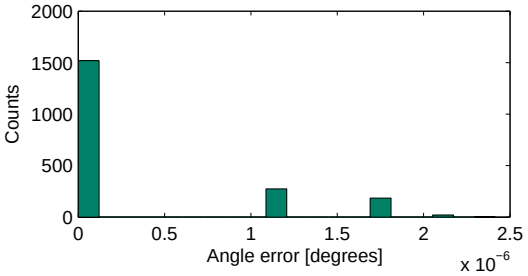
(a) Initial translation errors



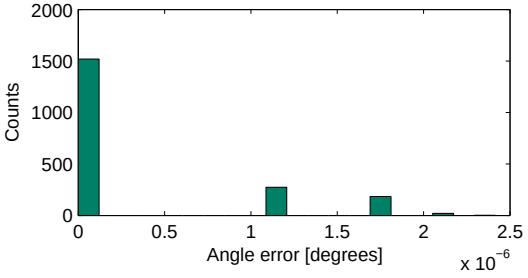
(b) Initial rotation errors



(c) Final rotation errors for the consensus in the manifold



(d) Final rotation errors for the consensus in the tangent space



Final rotation errors for the combined algorithm

Fig. 2. Simulation results (translation and rotation errors) with localized network

	Non-loc.	Loc.
C. in manifold (rot)	$7.19 \cdot 10^{-2}$	$5.08 \cdot 10^{-2}$
C. in tangent space (rot)	$1.71 \cdot 10^{-6}$	$2.41 \cdot 10^{-6}$
Combined (rot)	$1.99 \cdot 10^{-5}$	$1.01 \cdot 10^{-5}$
Translation	$1.87 \cdot 10^{-8}$	$2.04 \cdot 10^{-8}$

Table 1. Maximum errors for rotations and translations with and without localized network.

of the number of iterations. These results are shown in Fig. 3. We use a logarithmic scale for the y-axis in order to highlight the fact that final error is actually not zero. This is because the algorithm does not preserve the mean of the data from one iteration to the other. To better illustrate this last assertion, in Fig. 4, we plot the error between the mean of the $R_i^{(k)}$'s with the global algorithm at the beginning and after each iteration. As we can see, at the beginning such error is zero but it rapidly increases after few iterations.

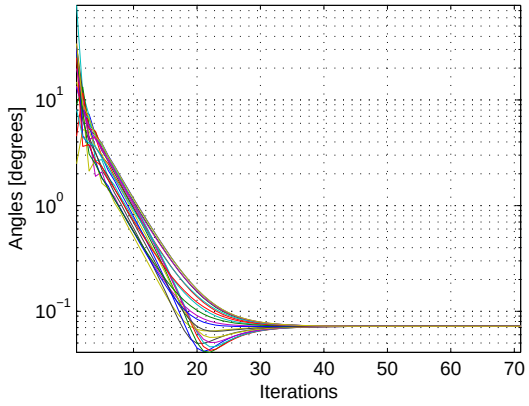


Fig. 3. Rotation errors with respect to the real mean

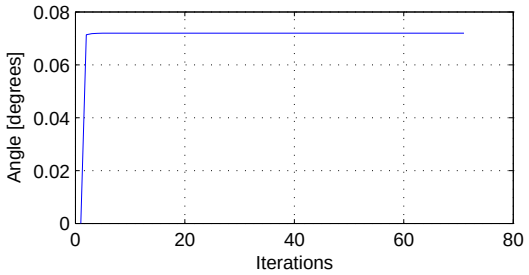


Fig. 4. Rotation errors with respect to the real mean

This is not the case for the consensus in the tangent space. The error between each node and the correct mean for this algorithm, with the same input data as before, is shown in Fig. 5. As expected, all the nodes have the same error (because their estimates are aligned). The error decreases in a small number of iterations (but remember that each iteration requires to run the Euclidean consensus in the tangent space

until convergence) and the final error is zero.

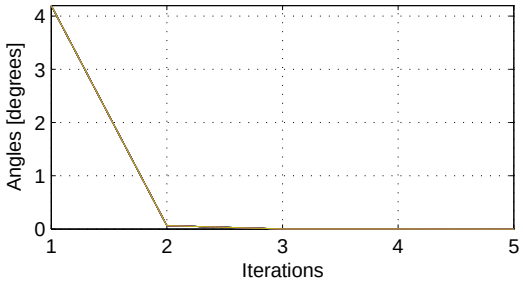


Fig. 5. Rotation errors with respect to the real mean

5. CONCLUSION AND FUTURE WORK

In this paper we considered the problem of distributed averaging of an object pose in a camera sensor network. Our algorithm combines distributed consensus and optimization on manifold. The main assumption we have done is that the network must be localized. We assessed the validity of our approach through synthetic simulations.

In our future work to we plan to apply our algorithm to real data (e.g., to estimate the pose of a face) and then we will try to extend our framework to the problem of network localization.