Distributed Synchronization of Euclidean Transformations with Guaranteed Convergence

Johan Thunberg[1*], Florian Bernard[2], Jorge Gonçalves[1]

***Abstract*— This paper addresses synchronization of Euclidean transformations over graphs. Synchronization in this context, unlike rendezvous or consensus, means that composite transformations over loops in the graph are equal to the identity. Given a set of non-synchronized transformations, the problem at hand is to find a set of synchronized transformations approximating well the non-synchronized transformations. This is formulated as a nonlinear least-squares optimization problem. We present a distributed synchronization algorithm that converges to the optimal solution to an approximation of the optimization problem. This approximation stems from a spectral relaxation of the rotational part on the one hand and from a separation between the rotations and the translations on the other. The method can be used to distributively improve the measurements obtained in sensor networks such as networks of cameras where pairwise relative transformations are measured. The convergence of the method is verified in numerical simulations.**

I. Introduction

We consider the following setting. There is a set of Euclidean coordinate systems corresponding to nodes in a connected graph. The coordinate systems are related by (pairwise) Euclidean transformations. Only a subset of those transformations are available and they label the edges of the connected graph. The transformations between the coordinate systems must be so-called *synchronized*, meaning that compositions of transformations over loops in the graph equal identity. This paper addresses the case where the relative transformations in the graph are not necessarily synchronized, i.e., compositions of transformations over loops are not necessarily equal to the identity. A distributed algorithm is developed for finding synchronized transformations approximating well the non-synchronized ones. In the algorithm, the orthogonal matrices and the translations—together comprising the Euclidean transformations—are computed concurrently.

Finding synchronized approximations of non-synchronized transformations is relevant to a wide range of applications. Examples of such include the 3D localization problem, where rigid transformations are calculated from camera measurements [1]–[3]; the problem of registering multiple images [4]; the Generalized Procrustes Problem, where rotations, translations and scales are calculated from multiple point-clouds [5]–[9]. Several more applications are enlisted in [10].

[1] Johan Thunberg and Jorge Gonçalves are with Luxembourg Centre for Systems Biomedicine (LCSB), 6, avenue du Swing, L-4367 Belvaux, Luxembourg.

[2] Florian Bernard is with Max Planck Institute for Informatics, Saarland Informatics Campus, Campus E1 4, 66123 Saarbrucken, Germany

*For correspondence, johan.thunberg@uni.lu. The authors gratefully acknowledge the financial support from Fonds National de la Recherche, Luxembourg (FNR8864515, FNR6538106).

Lately, the synchronization problem has been extensively studied; especially the case where the transformations are restricted to be orthogonal or rotations. For that case, Govindu et al. used Lie-group averaging, where a first-order approximation in the tangent space is employed [11]–[13]. In the works by of Singer et al., several optimization-based approaches were considered [14]–[16]. The same group of authors have presented several more application-oriented results [17]–[20]. Their work was later adapted by Pachauri et al. to the case where the transformations are permutation matrices [21].

In [22], [23] the related graph realization problem was solved. In that problem one shall find points in $\mathbb{R}^d$ corresponding to nodes in a sparsely connected graph, such that distances (scalar and positive) between the points correspond to given positive weights for the edges in the graph in the "best" way. A two-step procedure was proposed to solve the problem. In the first step, the eigenvector synchronization problem was solved [15] to find an orthogonal transformation. In the second step, a least-squares problem was solved to find translations. This procedure for solving the graph realization problem bears a resemblance to the method presented in this paper for the synchronization of Euclidean transformations. However, in our distributed method we concurrently solve for the rotations and the translations, where the rotations are solved by an extended spectral relaxation method and the translations are solved by a linear least-squares method.

Our method was developed with two objectives in mind. On the one hand it can be used to reduce computational speed; in each iteration it can be run in parallel. On the other hand it can be used in multi-agent systems, where agents (e.g. robots) equipped with cameras exchange local information—it can be used in applications where communication and sensor constraints comprise hurdles for a centralized solution to work satisfactorily. The proposed method is an extension of Algorithm 1 in [24]. That algorithm was designed for the special case of orthogonal transformations, whereas here we also consider translations.

In simulations, the algorithm convergences, up to a small gap, to a solution calculated with a Gauss-Newton method under an affine relaxation of the problem.

II. Problem formulation

A. Preliminaries

This work addresses synchronization of Euclidean transformations. When homogeneous coordinates are used, these

transformations are linear and represented by matrices. Formally, an element in $E(d)$—a Euclidean transformation—is a matrix

$$G = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix},$$

where $R \in O(d)$, $t \in \mathbb{R}^d$, and $1 \in \mathbb{R}$. The matrix group $O(d)$ is defined by

$$O(d) = \{Q \in \mathbb{R}^d : Q^T Q = I\}.$$

The inverse of G is

$$G^{-1} = \begin{bmatrix} R^T & -R^T t \\ 0 & 1 \end{bmatrix}.$$

We consider the following multi-agent system setting. There are n Euclidean coordinate systems, corresponding to n agents. We define the set $\mathcal{V} = \{1, 2, \ldots, n\}$. Henceforth we assume that the dimension d is larger than or equal to 2.

There is a directed, connected, and symmetric graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ describing the interaction between the agents. The neighbors to agent i are those j's contained in the set $\mathcal{N}_i = \{j : (i,j) \in \mathcal{E}\}$. Those are the agents that agent i interacts with.

Now, for each edge $(i,j) \in \mathcal{E}$, there is a corresponding $G_{ij} \in E(d)$. Formally we have at hand the tuple $(\mathcal{G}, g)$, where $g : \mathcal{E} \to E(d)$ is defined by $(i,j) \mapsto G_{ij}$ for all (i,j). Synchronization means that compositions of transformations over loops in the graph equals the identity, i.e.,

$$\begin{aligned} &G_{i_1 i_2} G_{i_2 i_3} \cdots G_{i_n i_i} = I \text{ if} \\ &\{(i_1, i_2), (i_2, i_3), \ldots, (i_n, i_1)\} \subset \mathcal{E}. \end{aligned} \tag{1}$$

Connectivity of the graph $\mathcal{G}$ is key here. If the G_{ij}'s are

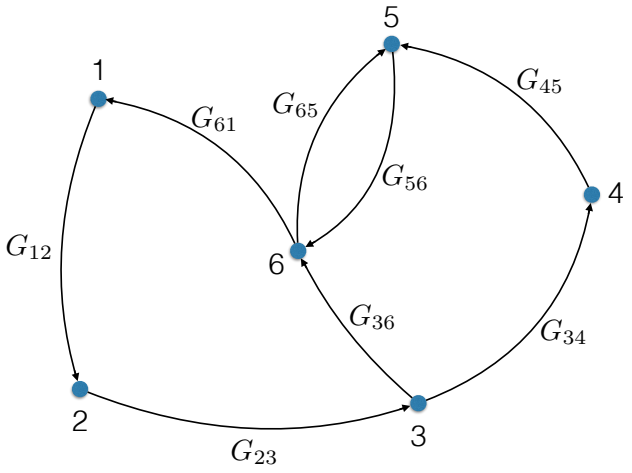

Fig. 1: Graphical illustration of a certain $(\mathcal{G}, g)$.

synchronized there is a unique (up to a global change of coordinates) set of G_i-transformations such that

$$G_{ij} = G_i^{-1} G_j \text{ for all } (i,j) \in \mathcal{E}. \tag{2}$$

if and only if $\mathcal{G}$ is connected [24]. Fig. 1 is a graphical illustration of $(\mathcal{G}, g)$ for a graph with six nodes.

B. A first problem

When the G_{ij}'s are not synchronized, the goal is to find—in a distributed way—G_i's such that the $(G_i^{-1} G_j)$-transformations are close to the G_{ij}'s for all $(i,j) \in \mathcal{E}$. By "close", we mean that the function f is minimized or is close to being minimized, where $f(G)$ is equal to

$$\frac{b_{ij}}{2} + f_1(R) + f_2(R,T) = \sum_{(i,j)\in\mathcal{E}} \|(G_{ij} - G_i^{-1} G_j) P_{ij}\|_F^2$$

where $P_{ij} = \frac{1}{\sqrt{2}} \text{diag}([\sqrt{a_{ij}}, \sqrt{a_{ij}}, \sqrt{a_{ij}}, \sqrt{b_{ij}}]^T)$ and

$$f_1(R) = \sum_{(i,j)\in\mathcal{E}} \frac{a_{ij}}{2} \|R_{ij} R_j^T - R_i^T\|_F^2, \tag{3}$$

$$f_2(R,T) = \sum_{(i,j)\in\mathcal{E}} \frac{b_{ij}}{2} \|t_{ij} - R_i^T (t_j - t_i)\|_2^2, \tag{4}$$

$$G = [G_1, G_2, \ldots, G_n]^T, \quad R = [R_1, R_2, \ldots, R_n]^T,$$
$$T = [t_1^T, t_2^T, \ldots, t_n^T],$$
$$G_i = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix} \text{ and } R_i \in O(d), t_i \in \mathbb{R}^d \text{ for all } i.$$

Beware of the difference between the variable T and the transpose operator $(\cdot)^T$. The parameters a_{ij} and b_{ij} are positive scalars for all $(i,j) \in \mathcal{E}$, which can be chosen to adjust for outliers or differences in the variance when the matrices are drawn from different distributions. The functions f, f_1 and f_2 are implicitly parameterized by $(\mathcal{G}, g)$, the a_{ij}'s, and the b_{ij}'s.

The problem we would like to solve in a distributed manner is

$$(P_1) \quad \begin{cases} \underset{G}{\text{minimize}} \ f(G), \\ G = [G_1, G_2, \ldots, G_n]^T, \\ \text{s.t. } G_i \in E(d) \text{ for all } i. \end{cases}$$

The solution to problem (P_1) is the (weighted) least-squares projection of the G_{ij}'s onto the set of synchronized transformations. For the translations, the choice of the Euclidean metric (see f_2) seems natural. Regarding f_1, there is a connection between the Frobenius norm and geodesics on on the rotational group in the special case of $SO(3)$. For $Q \in SO(3)$ it holds that $\|I - Q\|_{\text{F}}^2 = 8 \sin(\theta)$, where θ is the Riemannian distance (or angular distance), see [25]. Thus, for small errors, the minimizer of f_1 is close to the minimizer of a weighted least-squares problem for the Riemannian metric. It should aslo be mentioned here that the two metrics, i.e., the Frobenius norm and the Riemannian metric, are equivalent on $SO(3)$ in that they induce the same intrinsic metric [25].

C. A separation and a relaxation of the first problem (P_1)

Problem (P_1) is a non-convex (nonlinear) least squares problem over $(E(d))^n$. As such it is hard to find a global optimal solution. This motivates us to simplify the problem (P_1) into a form that is easier to solve. We will perform this simplification in two steps. The first is to separate the

minimization over the rotations and the translations, see problem (P_2) below.

$$(P_2) \quad \begin{cases} \underset{T}{\text{minimize}}\ f_2(R^*, T), \\ R^* \in \arg \min_{R \in (O(d))^n} f_1(R), \\ T = [t_1^T, t_2^T, \ldots, t_n^T]^T \in \mathbb{R}^{nd}. \end{cases}$$

In problem (P_2), $f_2(R^*, T)$ is a quadratic convex function in T. However, the minimization of f_1 with respect to R is still a non-convex problem. We will simplify this latter problem for the rotations by means of a relaxation to obtain a final problem (P_3), which is the problem for which we propose a distributed solution in the next section. The problem consists of two parts—the first is for the rotations and the second is for the translations.

$$(P_{3R}) \quad \begin{cases} \bar{R} \in \underset{R}{\text{argmin}}\ f_1(R), \\ R = [R_1, R_2, \ldots, R_n]^T \in \mathbb{R}^{nd \times d}, \\ R^T R = I_d, \\ R^* = [\text{Pr}_{O(d)}(\bar{R}_1), \text{Pr}_{O(d)}(\bar{R}_2), \ldots, \\ \text{Pr}_{O(d)}(\bar{R}_n)]^T, \end{cases}$$

$$(P_{3T}) \quad \begin{cases} \underset{T}{\text{minimize}}\ f_2(R^*, T), \\ T \in \mathbb{R}^{nd}, \\ R^* \text{ is obtained from } P_{3R}. \end{cases}$$

The problem (P_{3R}) can actually be seen as a procedure where we minimize f_1 over R, the columns of which are mutually orthogonal, in order to get a matrix $\bar{R}$, whose matrix-blocks are projected onto $O(d)$ to create R^*. The $\bar{R} = [\bar{R}_1, \bar{R}_2, \ldots, \bar{R}_n]^T$ and the R^* obtained in (P_{3R}) are unique up to a common orthogonal transformation from the right. The optimal value of (P_{3T}) is invariant to such a choice of a common orthogonal transformation.

(P_{3R}) is a spectral relaxation of the subproblem for R in the second line in (P_2). There are other relaxations such as semi-definite relaxations and least unsquared deviation combined with semi-definite relaxation [16]. An extensive treatment of the semi-definite relaxation approach can be found in [10]. On the positive side the semi-definite relaxation becomes linear for the our problem, on the negative side there does not seem to be available distributed methods.

III. The distributed algorithm

The distributed method we propose in this section solves the problem (P_3) under the assumption that $\mathcal{G}$ is symmetric (undirected) and connected. We recently proposed a distributed algorithm that solves subproblem (P_{3R}), see Algorithm 1 in [24]. The proposed method here is an extension of that method where problem (P_{3T}) is also solved distributively and concurrently with (P_{3R}). All the initializations and the updates for the orthogonal matrices and the associated auxiliary variables are the same as in our previous journal paper. Furthermore, we refer the reader to Section 4.1 in that paper for a detailed description of this part of the algorithm. Here we only provide a brief description.

The updates-section for the orthogonal matrices contains one basic part and one advanced part. The basic part comprises the updates for the $\tilde{R}_i(k)$'s and the $R_i(k)$'s and the advanced part contains the rest. The updates in the basic part are essentially gradient descent steps of the function f_1 and projections onto $O(d)$. The problem here is that the $\tilde{R}_i(k)$-matrices converge to zero on the one hand, and become more and more ill-conditioned on the other. The idea with the advanced part of the algorithm is to create weighted $\tilde{Q}_i(k)$-matrices (and their corresponding projected $Q_i(k)$-matrices) where the ill-conditioning has been compensated for by weighting-matrices. Those, in turn, are concurrently calculated via a distributed algorithm.

The (extended) proposed algorithm is:

Algorithm 1 Distributed method for synchronization of Euclidean transformations over symmetric graphs

---

Inputs: a symmetric directed graph $\mathcal{G} = (\mathcal{E}, \mathcal{V})$, a weight matrix $A = [a_{ij}]$, a weight matrix $B = [b_{ij}]$, and a collection $\{G_{ij}\}_{(i,j)\in\mathcal{E}}$ of matrices in $E(d)$. Each G_{ij} contains an orthogonal matrix R_{ij} and a translation vector t_{ij};

$$G_{ij} = \begin{bmatrix} R_{ij} & t_{ij} \\ 0 & 1 \end{bmatrix} \text{ for all } (i,j) \in \mathcal{E}.$$

Outputs: $\tilde{R}_i(k)$, $R_i(k)$, $\tilde{Q}_i(k)$, $Q_i(k)$, and $t_i(k)$ for $i \in \mathcal{V}$ and $k \geq 1$.

Initialization: let $\tilde{R}_i(0) = I_d$, $d_{is}(0) = 1$, $\tilde{d}_{is}(0) = 1$, and $\tilde{d}_{is}(-1) = 1$ for all i, s. Let $V_{ij} = (a_{ij} + a_{ji})I$ and $Q_{ij} = a_{ij}R_{ij} + a_{ji}R_{ji}^T$ for all $(i,j) \in \mathcal{E}$. Let $\epsilon_1, \epsilon_2 > 0$. Let $t_i(0) = 0$ for all i.

Iteration $k \geq 1$**:**

Updates for the orthogonal matrices

for all i, let

$$\begin{aligned} \tilde{R}_i(k) &= \tilde{R}_i(k-1) \\ &\quad + \epsilon_1 \sum_{j \in \mathcal{N}_i} (Q_{ij}\tilde{R}_j(k-1) - V_{ij}\tilde{R}_i(k-1)), \\ R_i^T(k) &= \text{Pr}_{O(d)}(\tilde{R}_i(k)), \\ \tilde{d}_{is}(k) &= \{ \text{ calculated in Subroutine 1 in [24] } \}, \\ d_{is}(k) &= d_{is}(k-1) + (\tilde{d}_{is}(k-1) - \tilde{d}_{is}(k-2)) \\ &\quad + \epsilon_2 \sum_{l \in \mathcal{N}_i} (d_{ls}(k-1) - d_{is}(k-1)) \\ &\quad \text{for } s = 1, 2, \ldots, d, \\ D_i(k) &= \text{diag}(d_{i1}(k), d_{i2}(k), \ldots, d_{id}(k)), \end{aligned}$$

$$\begin{aligned} \tilde{Q}_i(k) &= \{ \text{ calculated in Subroutine 1 in [24] } \}, \\ Q_i^T(k) &= \text{Pr}_{O(d)}(\tilde{Q}_i(k)(D_i(k))^{-\frac{1}{2}}), \end{aligned}$$

Updates for the translations

$$t_i(k+1) = t_i(k) + \epsilon_1 \sum_{j\in\mathcal{N}_i} (b_{ij} + b_{ji})(t_j(k) - t_i(k)) \\ -\epsilon_1 \sum_{j\in\mathcal{N}_i} (b_{ij}Q_i(k)t_{ij} - b_{ji}Q_j(k)t_{ji}).$$

The projection $\Pr_{O(d)}(\tilde{R}_i)$ onto $O(d)$ of the matrix $\tilde{R}_i(k)$ (there is also a projection of $\tilde{Q}_i(k)$), is defined in the least-squares sense [26] by $\Pr_{O(d)}(\tilde{R}_i) = UV^T$, where U and V are the left and right orthogonal matrices, respectively, given by the singular value decomposition (SVD) of $\tilde{R}_i$. If we consider the case of rigid transformations in 3D, the orthogonal transformations are restricted to the special orthogonal group, $SO(3)$. Then the projection onto $SO(3)$ is given by $\Pr_{SO(3)}(\tilde{R}_i) = U\,\mathrm{diag}(1, 1, \det(V^TU))V^T$.

Remark 1. In the updates for the translations in Algorithm 1, one can replace $Q_i(k)$ and $Q_j(k)$ by $R_i(k)$ and $R_j(k)$. By doing so, computational time is saved since we only need the first two lines in the update-part for the rotations—those that are needed for the calculation of the $R_i(k)$'s. In numerical simulations performance is comparable to, and the convergence speed is even faster compared to when the $Q_i(k)$'s and the $Q_j(k)$'s are used. However, by doing this "replacement", we loose the theoretical convergence guarantees that are provided in the next section.

IV. Convergence of Algorithm 1

The updates for the translations and those for the orthogonal matrices corresponding to the $\tilde{R}_i(k)$'s can be understood as gradient descent steps. The basic gradient descent approach for the orthogonal matrices have then been augmented by the introduction of a collection of auxiliary variables. With those, under some technical conditions (generally fulfilled and) provided in Table 1 below, we achieve convergence to the orthogonal projections of the matrices in the solution to the spectral relaxation (P_{3R}).

(1)	$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is connected and symmetric.
(2)	$R_{ij} \in O(d)$ for all $(i, j) \in \mathcal{E}$.
(3)	$\epsilon_1 < \frac{2}{\|P\|_2}$, where $P = \mathrm{diag}((A + A^T)1_n) + A + A^T$.
(4)	$R_i \in GL(d, \mathbb{R})$ for all the R_i-matrices in X.
(5)	$\sum_{i\in\mathcal{V}} R_i \in GL(d, \mathbb{R})$, where the R_i are the matrices in $\bar{R}$ in (P_{3R}).
(6)	It holds that $\lambda_{(n-1)d} > \lambda_{(n-1)d+1}$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ are the eigenvalues of L_{undir}, i.e., $L_{\text{undir}} = V\mathrm{diag}([\lambda_1, \lambda_2, \ldots, \lambda_{nd}])V^T$ where L_{undir} is the Hessian matrix of f_1 with respect to R.
(7)	It holds that $\lambda_{nd-(i+1)} > \lambda_{nd-i}$, for $i = 0, 1, \ldots d-2$ where the λ_i are defined in (8) above.
(8)	$\epsilon_2 < \frac{2}{\|L\|_2}$, where L is the graph Laplacian matrix of the graph $\mathcal{G}$ when assuming all edge-weights are equal to 1.

TABLE I: Conditions for convergence.

We begin by recalling the following result from [24], where it is presented by the two propositions 13 and 14.

Proposition 1 ([24]). *Suppose that the convergence conditions (1-8) in Table 1 are satisfied. Then, for Algorithm 1 there is a positive integer K such that $\tilde{R}_i^{-1}(k)$ is well defined for all i and $k \geq K$, and (for $k \geq K$) it holds that*

$$(\tilde{R}_i(k)\tilde{R}_j^{-1}(k) \to \bar{R}_i\bar{R}_j^{-1} \text{ as } k \to \infty) \text{ for all } (i, j) \in \mathcal{E}, \tag{5}$$

$$\tilde{Q}(k) \to \bar{R} \text{ as } k \to \infty, \tag{6}$$

$$Q(k) \to R^* \text{ as } k \to \infty, \tag{7}$$

where $\bar{R}$ and R^* are obtained from problem (P_{3R}) (up to a common orthogonal transformation from the right) and the matrices $\tilde{Q}(k)$ and $Q(k)$ are defined as $\tilde{Q}(k) = [\tilde{Q}_1(k), \tilde{Q}_2(k), \ldots, \tilde{Q}_n(k)]^T$ and $Q(k) = [Q_1(k), Q_2(k), \ldots, Q_n(k)]^T$, respectively.

A more detailed description of the convergence conditions in Table 1 can be found in [24]. Conditions (1), (2) are self explanatory. Conditions (3), (8) are bounds for the step sizes ϵ_1 and ϵ_2. Conditions (5) is an assumption about the invertability of the matrices in $\bar{R}$ in (P_{3R}); these matrices will always be invertible when the R_{ij}'s are sufficiently close to being synchronized. Conditions (6)–(7) are assumptions about the Hessian matrix of f_1 computed with respect to R. The important thing here is that these assumptions are in general satisfied. In Proposition 1 there are three different convergence results provided by (5),(7).

The first, (5), states that the relative transformations, obtained by using the $\tilde{R}_i(k)$'s, asymptotically converge to those obtained by using the $\bar{R}_i$'s, which is the optimal "non-projected" solution of (P_{3R}). Now, interesting in its own right, this result is not very helpful as neither the $\bar{R}_i$'s nor the $(\bar{R}_i^T\bar{R}_j)$'s are necessarily orthogonal, which we require.

The second and the third results, (6),(7), are more important. They state that the $\tilde{Q}(k)$'s and the $Q(k)$'s converge to optimal solutions and "projected optimal solutions" to (P_{3R}), respectively. It should be noted that there are infinitely many optimal solutions equivalent up to orthogonal transformation. Furthermore, the results are somewhat stronger than those presented in [24] in that we guarantee convergence to points and exclude the possibility of limit cycles in the set of optimal solutions. The formulations in [24] were unnecessarily conservative in this regard. The result (7) implies that

$$(Q_i^T(k)Q_j(k) \to (\Pr_{O(d)}(\bar{R}_i))(\Pr_{O(d)}(\bar{R}_j))^T \text{ as } k \to \infty) \\ \text{for all } (i, j) \in \mathcal{E}. \tag{8}$$

Now we turn to the translations and provide the following result.

Proposition 2. *Suppose that the convergence conditions (1-8) in Table 1 are satisfied. Then, for Algorithm 1 there is R^* obtained from problem (P_{3R}) (up to a common orthogonal transformation from the right), a positive integer K such that $\tilde{R}_i^{-1}(k)$ is well defined for all i and $k \geq K$, and (for $k \geq K$) it holds that $T(k)$ converges to the set of optimal solutions to (P_{3T}) for the given R^*.*

Proof. When R^* is fixed, the Hessian matrix of f_2 computed with respect to T is the same as a the Hessian matrix of f_1 computed with respect to R under the assumption that the R_{ij}'s are equal to I. If we were to replace $Q(k)$ by R^* in the updates for the translations in Algorithm 1, convergence condition (3) in Table 1 would guarantee that the updates amounted to gradient descent steps for $f_2(R^*, T)$. Convergence to the set of minimizers of f_2 would be guaranteed.

Now, it is not the case that $Q(k) = R^*$, rather $Q(k)$ converges to R^* as k goes to infinity (up to orthogonal transformation), see Proposition 1. The rest of the proof addresses the possibility that problems might arise due to the transient-behavior when $Q(k)$ converges to R^*.

We can write f_2 as $f_2(R(k), T) = \frac{1}{2}T^T H_T T + c(Q(k))^T T$, where H_T is the Hessian matrix and, and $c(Q(k))$ is a vector, which is a linear function of $Q(k)$ (once again, beware of the difference between the transpose symbol T and the variable T). Due to convergence condition (3), in each iteration the updates for the translations comprise a gradient descent step for f_2 when $Q(k)$ is regarded as fixed (see discussion in the first paragraph of the proof above), i.e.,

$$f_2(Q(k), T(k+1)) < f_2(Q(k), T(k)) \text{ for all } k \quad (9)$$

if $T(k)$ is not a minimizer of f_2 with the chosen $Q(k)$. Let the set of optimal solutions for f_2 with respect to T and with the chosen $Q(k)$ be $\mathcal{T}(Q(k)) = T_k^* + \ker(H_T)$, where T_k^* is the unique vector contained in $\text{im}(H_T)$ that satisfies $H_T T_k^* = -c(Q(k))$. Let us split $T(k)$ into two orthogonal parts corresponding to $\text{im}(H_T)$ and $\ker(H_T)$, respectively; $T(k) = T_{\text{im}}(k) + T_{\ker}(k)$ for all k. Let $T_{\text{im}}(k) = Bv(k)$, where the columns of B comprise an orthonormal base for $\text{im}(H_T)$. For T_k^*, there is a corresponding v_k^* given explicitly by $v_k^* = -(B^T H_T B)^{-1} B^T c(Q(k))$. Thus $T_k^* = -B(B^T H_T B)^{-1} B^T c(Q(k))$.

The updates for $T(k)$ are given by

$$T(k+1) = T(k) - \epsilon_1 (H_T T(k) + c(Q(k))). \quad (10)$$

By using (10) and the fact that the columns in B are orthonormal we obtain that

$$\begin{aligned} v(k+1) &= v(k) - \epsilon_1 B^T H_T B v(k) - \epsilon_1 B^T c(k) \text{ or} \\ v(k+1) - v_k^* &= (I - \epsilon_1 B^T H_T B)(v(k) - v_k^*). \end{aligned} \quad (11)$$

The $Q(k)$'s converge and v_k^* is a linear function of $Q(k)$. This means that there is v^* such that $v_k^* \to v^*$ as $k \to \infty$. Let us introduce $z(k) = v(k) - v^*$ and $u(k) = v^* - v_k^*$. We rewrite (11) with the new variables:

$$z(k+1) = (I - \epsilon_1 B^T H_T B) z(k) - \epsilon_1 B^T H_T B u(k). \quad (12)$$

Now, due to convergence condition (3) in Table 1 and the fact that $u(k)$ is bounded and goes to zero as k goes to infinity, we can conclude that $z(k)$ goes to zero as k goes to infinity. This in turn means that $v(k)$ converges to v^*, which in turn means that $T(k)$ converges to $\mathcal{T}(R^*)$. □

V. Numerical experiments

To evaluate the performance of Algorithm 1, numerical simulations were conducted. Results are shown in Fig. 2. For the two different parameter settings considered—differing by the choice of graph density—100 simulations were run. All the simulations were for $SE(3)$. Similar convergence results were also obtained for $E(3)$ and higher dimensions.

Fig. 2, shows the gap in percent between the objective value for Algorithm 1 at each iteration and that of a Gauss-Newton method. The number of iterations in the Gauss-Newton method was chosen to five, even though the main convergence was observed in the simulations after at most three iterations. The Gauss-Newton method can be found, up to the a_{ij} and the b_{ij} parameters, in sections 3.7 and 3.8 in [27]. The method is used to calculate a local optimum (under a relaxation of the constraints, see below) of (P_1), where the solution to (P_3) has been used as initialization. For the Gauss-Newton method, the constraints on the R_i-matrices are relaxed in the following way. They are only required to be invertible, implying that the G_i-matrices are only required to be affine. Thus, the objective function value of the local optimum obtained via the Gauss-Newton method is an underestimate of the objective function value of the closest local optimum (closest to the solution obtained by Algorithm 1) obtained without the relaxations of the R_i's.

The solid lines in the plots in Fig. 2 are the mean gap over the 100 simulations as functions of the iterations in the algorithm. The dashed lines illustrate mean square deviations above and below the mean. In all simulations the step sizes ϵ_1 and ϵ_2 were chosen to $1/(2n)$. The other parameters were defined as follows: n is the number of coordinate systems and $d = 3$; ρ is the graph density. We always assume that the graphs are connected. Thus, unlike the classical definition, $\rho = 0$ corresponds to a tree graph and $\rho = 1$ corresponds to the complete graph, with linear interpolation in between; σ_R is the standard deviation in the generation of the non-synchronized rotations; σ_T is the standard deviation in the generation of the inconsistent translations; σ_a is width of the support region in the generation of the a_{ij}'s; σ_b is width of the support region in the generation of the b_{ij}'s;

Each simulation for each setting of parameters was conducted in the following way.

1) A set of n rigid transformations are constructed. The rotations are created by first generating matrices whose elements are drawn from the uniform distribution with the interval $(-0.5, 0.5)$ as support. Those matrices are then projected onto $SO(3)$ to generate the rotation matrices. The translations are vectors whose elements are drawn from the uniform distribution with the interval $(-0.5, 0.5)$ as support. **2)** A set of n^2 synchronized rigid transformations were generated from the transformations generated in step 1 according to eq. (2) for all (i, j)-pairs. **3)** n^2 number of non-synchronized transformations are generated from the n^2 number of transformations generated in step 2. For each rotation matrix, a new matrix is first generated by element-wise addition of Gaussian noise with standard deviation σ_R.

Then, that matrix is projected back onto $SO(3)$. All such projections are always done in the least-squares sense by means of the singular value decomposition. For each translation vector, a new translation vector is generated by element-wise addition of Gaussian noise with standard deviation σ_T. **4)** The graph, the a_{ij}'s and the b_{ij}'s are generated. The latter parameters are drawn from the uniform distribution over $[1-\sigma_a, 1]$ and $[1-\sigma_b, 1]$ respectively. We let $a_{ij} = a_{ji}$. **5)** The methods are evaluated for the transformations and the parameters generated in the first four steps.

An important factor when it comes to the convergence rate is the connectivity of the graph. Another phenomenon that is not captured by the simulations, is that the performance and stability of the algorithm is independent of the size of the norms of the t_{ij} vectors. This is a consequence of the fact that the matrix H_T is not a function of the t_{ij}.

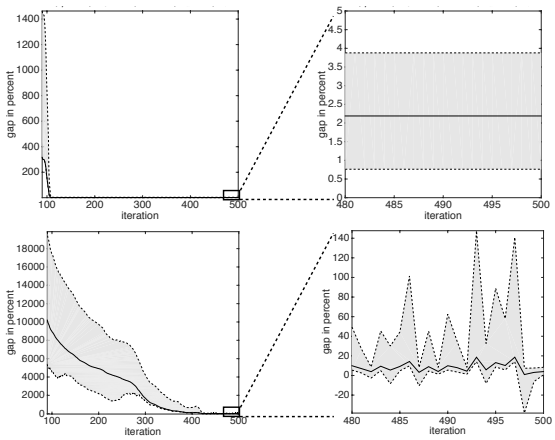

Fig. 2: Convergence of our algorithm for two different parameter settings. In the top plots $\rho = 0.8$ and in the bottom plots $\rho = 0.4$. The other parameters were chosen to $n = 50$, $\sigma_R = 0.01$, $\sigma_T = 0.2$, $\sigma_a = 0.1$, and $\sigma_b = 0.1$.

VI. Conclusions

In this paper we presented a distributed method for synchronizing inconsistent Euclidean transformations over graphs. We prove convergence to an approximation of the nonlinear synchronization problem. In the proposed algorithm, the orthogonal matrices and the translation vectors are synchronized concurrently. Numerical simulations show the convergence of the algorithm. In all simulations, the transformations converge close to critical points calculated with a Gauss-Newton method for an affine relaxation of the synchronization problem.

References

[1] R. Aragues, C. Sagues, and Y. Mezouar, *Parallel and Distributed Map Merging and Localization: Algorithms, Tools and Strategies for Robotic Networks*. Springer, 2015.

[2] R. Tron and R. Vidal, "Distributed 3-d localization of camera sensor networks from 2-d image measurements," *Transactions on Automatic Control*, vol. 59, no. 12, pp. 3325–3340, 2014.

[3] E. Montijano, D. Zhou, M. Schwager, and C. Sagues, "Distributed formation control without a global reference frame," in *American Control Conference (ACC), 2014*. IEEE, 2014, pp. 3862–3867.

[4] F. Bernard, J. Thunberg, A. Husch, L. Salamanca, P. Gemmar, F. Hertel, and J. Goncalves, "Transitively consistent and unbiased multi-image registration using numerically stable transformation synchronisation," in *MICCAI Workshop on Spectral Analysis in Medical Imaging (SAMI)*, 2015.

[5] K. Arun, T. Huang, and S. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 5, pp. 698–700, 1987.

[6] J. Gower and G. Dijksterhuis, *Procrustes problems*. Oxford University Press Oxford, 2004, vol. 3.

[7] P. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, Mar. 1966.

[8] B. Horn, H. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society of America A*, vol. 5, no. 7, p. 1127, 1988.

[9] F. Bernard, J. Thunberg, P. Gemmar, F. Hertel, A. Husch, and J. Goncalves, "A Solution for Multi-Alignment by Transformation Synchronisation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[10] N. Boumal, "A riemannian low-rank method for optimization over semidefinite matrices with block-diagonal constraints," *arXiv preprint arXiv:1506.00575*, 2015.

[11] V. Govindu, "Lie-algebraic averaging for globally consistent motion estimation," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR.* IEEE, 2004.

[12] ——, "Robustness in motion averaging," in *Computer Vision–ACCV 2006*. Springer, 2006, pp. 457–466.

[13] V. Govindu and A. Pooja, "On averaging multiview relations for 3d scan registration," *Transactions on Image Processing*, vol. 23, no. 3, pp. 1289–1302, 2014.

[14] A. Bandeira, A. Singer, and D. Spielman, "A cheeger inequality for the graph connection laplacian," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 4, pp. 1611–1630, 2013.

[15] A. Singer, "Angular synchronization by eigenvectors and semidefinite programming," *Applied and computational harmonic analysis*, vol. 30, no. 1, pp. 20–36, 2011.

[16] L. Wang and A. Singer, "Exact and stable recovery of rotations for robust synchronization," *Information and Inference*, p. iat005, 2013.

[17] K. Chaudhury, Y. Khoo, and A. Singer, "Global registration of multiple point clouds using semidefinite programming," *arXiv.org*, Jun. 2013.

[18] R. Hadani and A. Singer, "Representation theoretic patterns in three dimensional Cryo-Electron Microscopy I: The intrinsic reconstitution algorithm," *Annals of mathematics*, vol. 174, no. 2, p. 1219, 2011.

[19] ——, "Representation Theoretic Patterns in Three-Dimensional Cryo-Electron Microscopy II—The Class Averaging Problem," *Foundations of computational mathematics (New York, N.Y.)*, vol. 11, no. 5, pp. 589–616, 2011.

[20] A. Singer and Y. Shkolnisky, "Three-Dimensional Structure Determination from Common Lines in Cryo-EM by Eigenvectors and Semidefinite Programming," *SIAM journal on imaging sciences*, vol. 4, no. 2, pp. 543–572, Jun. 2011.

[21] D. Pachauri, R. Kondor, and V. Singh, "Solving the multi-way matching problem by permutation synchronization," in *Advances in neural information processing systems*, 2013, pp. 1860–1868.

[22] M. Cucuringu, Y. Lipman, and A. Singer, "Sensor network localization by eigenvector synchronization over the euclidean group," *ACM Transactions on Sensor Networks (TOSN)*, vol. 8, no. 3, p. 19, 2012.

[23] M. Cucuringu, A. Singer, and D. Cowburn, "Eigenvector synchronization, graph rigidity and the molecule problem," *Information and Inference*, vol. 1, no. 1, pp. 21–67, 2012.

[24] J. Thunberg, F. Bernard, and J. Goncalves, "Distributed methods for synchronization of orthogonal matrices over graphs," *Automatica, to appear*, 2017.

[25] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International journal of computer vision*, vol. 103, no. 3, pp. 267–305, 2013.

[26] J. Keller, "Closest unitary, orthogonal and hermitian operators to a given operator," *Mathematics Magazine*, vol. 48, no. 4, pp. 192–197, 1975.

[27] J. Thunberg, F. Bernard, and J. Goncalves, "On transitive consistency for linear invertible transformations between euclidean coordinate systems," *arXiv preprint arXiv:1509.00728*, 2015.