Robust Group Synchronization via Quadratic Programming

Yunpeng Shi [*1] **Cole Wyeth** [*2] **Gilad Lerman** [2]

Abstract

We propose a novel quadratic programming formulation for estimating the corruption levels in group synchronization, and use these estimates to solve this problem. Our objective function exploits the cycle consistency of the group and we thus refer to our method as detection and estimation of structural consistency (DESC). This general framework can be extended to other algebraic and geometric structures. Our formulation has the following advantages: it can tolerate corruption as high as the information-theoretic bound, it does not require a good initialization for the estimates of group elements, it has a simple interpretation, and under some mild conditions the global minimum of our objective function exactly recovers the corruption levels. We demonstrate the competitive accuracy of our approach on both synthetic and real data experiments of rotation averaging.

1. Introduction

Group synchronization (GS) is a critical mathematical problem that has broad applications in statistics and computer science. It assumes a mathematical group $\mathcal{G}$ and a graph $G([n], E)$ where $[n] = \{1, 2, \dots, n\}$ and E is the set of edges. Each node i of the graph is assigned an unknown group element g_i^*, where the star superscript designates ground-truth information. At each edge $ij \in E$, one observes noisy and corrupted measurements g_{ij} of the ground-truth group ratio $g_{ij}^* = g_i^* g_j^{*-1}$. The GS problem asks to recover the unknown group elements, $\{g_i^*\}_{i\in[n]}$, from the observed group ratios, $\{g_{ij}\}_{ij\in E}$. The most well-known group synchronization problem is rotation averaging in 3D computer vision, where $\mathcal{G}$ is $SO(3)$. It asks to recover the absolute rotations of objects from the possibly corrupted and noisy relative rotations between pairs of objects. The rotation averaging problem plays a central role in many 3D computer vision tasks such as structure from motion (SfM) and simultaneous localization and mapping (SLAM). Other examples of group synchronization include phase synchronization ($\mathcal{G} = SO(2)$) with applications to cryo-electron microscopy imaging, permutation synchronization ($\mathcal{G} = S_n$) with applications to multi-object matching, and $\mathbb{Z}_2$-synchronization ($\mathcal{G} = \mathbb{Z}_2$) with applications to correlation clustering and community detection.

In many real scenarios, the observed group ratios are highly corrupted. In order to handle the high corruption, it is critical to estimate the corruption levels. In order to quantify them we assume a predefined bi-invariant metric on $\mathcal{G}$, which we denote by d. The bi-invariance property of d means that for any $g_1, g_2, g_3 \in \mathcal{G}$, $d(g_1, g_2) = d(g_3 g_1, g_3 g_2) = d(g_1 g_3, g_2 g_3)$. Using d, the corruption level of edge $ij \in E$ is

$$s_{ij}^* = d(g_{ij}, g_{ij}^*). \tag{1}$$

The primary goal of this work is to develop a principled and interpretable framework to estimate $\{s_{ij}^*\}_{ij\in E}$ without requiring a good initialization, and to robustly estimate the group elements using the estimated $\{s_{ij}^*\}_{ij\in E}$.

For this purpose we exploit the cycle-consistency constraint. It uses the group identity e as follows:

$$g_{ij}^* g_{jk}^* g_{ki}^* = e \text{ for any } ij, jk, ki \in E, \tag{2}$$

In principle, one can utilize the cycle-consistency constraints for longer cycles, whereas we focus on 3-cycles for simplicity and for computational efficiency.

We propose a general solution for GS, which we refer to as Detection and Estimation of Structural Consistency (DESC). The terminology "structural consistency" refers to the induced cycle consistency constraint. We believe that this framework can also be generalized to other algebraic or geometric structure constraints in other application areas, such as low-rankness, low dimensionality, coplanarity of points in a Euclidean space, transitivity of ordering in the ranking problem, etc. Nevertheless, in order to keep this presentation focused and clear, we restrict it to GS. Similarly, we discuss the most refined procedures of DESC for the group $SO(3)$ only.

1.1. Previous Works

Group synchronization was commonly applied to the discrete groups $\mathbb{Z}_2$ and S_n (Bandeira, 2018; Cucuringu,

[*]Equal contribution [1]Program in Applied and Computational Mathematics, Princeton University [2]School of Mathematics, University of Minnesota. Correspondence to: Yunpeng Shi <yunpengs@princeton.edu>, Cole Wyeth <wyeth008@umn.edu>, Gilad Lerman <lerman@umn.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

2015; Ling, 2020; Chen et al., 2014; Pachauri et al., 2013) and the Lie group $SO(d)$ (Singer, 2011; Wang & Singer, 2013; Eriksson et al., 2018). Its most common formulation uses least squares (LS) minimization:

$$g_i = \arg\min_{g_i \in \mathcal{G}} \sum_{ij \in E} d^2(g_i g_j^{-1}, g_{ij}). \tag{3}$$

Since the domain of this minimization is a nonconvex group, it is often relaxed to an eigendecomposition problem (Cucuringu, 2015; Singer, 2011; Pachauri et al., 2013; Ling, 2020), or a semidefinite programming (SDP) problem (Bandeira, 2018; Singer, 2011; Chen et al., 2014), or when $\mathcal{G}$ is a Lie group it is solved locally and iteratively by tangent space approximations of the manifold (Govindu, 2004). For discrete groups, such as $\mathbb{Z}_2$ and S_n, this formulation is robust to corruptions. However, when $\mathcal{G}$ is a Lie group, such as $\mathcal{G} = SO(d)$, then it is sensitive to outliers. In this case, robustness to outliers can be achieved by either introducing a robust objective function, or applying an outlier detection algorithm to preprocess the corrupted data.

A common robust formulation is ℓ_p-minimization with $0 < p \leq 1$ (Wang & Singer, 2013; Chatterjee & Govindu, 2017; Hartley et al., 2011), where d^2 in (3) is replaced with d^p. It is often solved by the iteratively reweighted least squares (IRLS). The main limitation of IRLS is that it requires good initialization of group elements and may easily get stuck at local minima in the presence of high noise and corruption. A recent work by Maunu & Lerman (2020) uses an energy based on Tukey depth to achieve provable robust synchronization to arbitrary outliers, but it also requires local initialization.

Outlier detection methods for GS utilize cycle-consistency constraints to distinguish between clean and corrupted edges. In particular, Zach et al. (2010) suggested two methods, based on belief propagation and linear programming. Agarwal et al. (2020) proposed a similar linear programming approach for the different ranking problem. Shen et al. (2016) classified an edge as clean according to its appearance in a consistent cycle (note that such an approach cannot handle self-consistent corruption). We remark that both IRLS and outlier detection methods eventually rely on accurate assignment of edge weights. For outlier detection, the edge weights are binary, where zero weight corresponds to removing an edge. However, the binary weights do not exactly reflect the corruption levels and may thus result in suboptimal estimates for group elements. IRLS updates the edge weights from the estimated corruption levels. However, the corruption levels are heuristically estimated in an iterative procedure, which is sensitive to the initialized estimates of the group elements.

The recent cycle-edge message passing (CEMP) (Lerman & Shi, 2021) overcomes the aforementioned drawbacks of both IRLS and outlier detection methods. It estimates the corruption levels without requiring a good initialization or solving weighted least squares, even when the corruption is high. Given a set of 3-cycles, for each 3-cycle ijk, CEMP first computes the cycle inconsistency: $d_{ij,k} = d(g_{ij}g_{jk}g_{ji}, e)$. It then estimates the edge corruption level from those cycle inconsistencies via message passing between cycles and edges. However, the message passing procedure is hard to interpret as it does not explicitly optimize an objective function. Moreover, both its performance and theory rely on a set of reweighting parameters. A recent message passing least squares (Shi & Lerman, 2020) framework combines CEMP-like iterations with IRLS and achieves superior performance on a variety of datasets. However, it remains heuristic and lacks theoretical guarantees.

Few previous works for GS (Birdal et al., 2018; Sun et al., 2019; Birdal et al., 2020) further estimate the distribution of the group elements. They aim to address scene ambiguities and for this purpose assume special probabilistic models. Our proposed work estimates the distribution of corruption levels, assuming that a deterministic condition holds. This estimated distribution is merely used to improve the estimation of the corruption levels, though it might be used in the future for statistical inference and uncertainty quantification.

A common theoretical setting in GS assumes the uniform corruption model (UCM). In this model, the graph G is generated by the Erdős-Rényi model $G(n, p)$, where p is the probability of connecting two nodes. An edge is then independently corrupted with probability q. If $ij \in E$ is corrupted, then g_{ij} is i.i.d. sampled from a Haar measure on $\mathcal{G}$, otherwise, $g_{ij} = g^*_{ij}$. Under UCM, the information theoretic sample complexity for the exact recovery of group elements is $n/\log n = O(p^{-1}(1-q)^{-2})$ (Chen et al., 2016). For $\mathbb{Z}_2$ and S_n-synchronization, spectral and SDP methods match this bound (Cucuringu, 2015; Bandeira, 2018; Chen et al., 2014; Ling, 2020). However, for Lie group synchronization, it is a challenging open problem to prove that an algorithm can match the information theoretic sample complexity. The best sample complexity bound for $SO(2)$ and $SO(3)$ was established for CEMP: $n/\log n = O(p^{-2}(1-q)^{-8})$ (Lerman & Shi, 2021), but there is a clear gap with the desired bound.

1.2. This Work

In view of the limitations of the previous methods, we summarize the contributions of our proposed DESC.

- DESC is a novel quadratic programming framework for estimating the corruption levels of edges in GS. Its minimization formulation provides a simple interpretation. We prove that under mild conditions, the global minimum recovers the corruption levels
- We show that under UCM, the sample complexity of our QP formulation is $n/\log n = O(p^{-2}(1-q)^{-2})$. It matches the dependence of the information-theoretic bound on q, unlike previous methods
- Our QP formulation is parameter free and does not require

good initialization even in highly corrupted scenarios. We demonstrate that a naive projected gradient descent is able to obtain satisfying corruption estimates
- For rotation averaging, we develop an algorithm for estimating the absolute rotations using the corruption levels that are estimated from our QP framework (we refer to both the latter framework and the former algorithm as DESC). Both synthetic and real data experiments demonstrate the state-of-the-art accuracy of this algorithm

The rest of the paper is organized as follows: §2 presents the framework of DESC for GS, its theoretical guarantees and the DESC algorithm for rotation averaging; §3 tests DESC on synthetic and real datasets; and §4 concludes this work. The supplemental code is in https://github.com/ColeWyeth/DESC

2. The DESC Framework

We explain DESC as follows: §2.1 introduces notation and preliminary observations; §2.2 and §2.3 formulate and motivate the DESC framework for corruption estimation; §2.4 demonstrates the theory of this formulation under UCM; §2.5 describes the optimization method that we adopted to solve this formulation; §2.6 clarifies its computational complexity; §2.7 explains how to generally recover group elements using the output of our framework; and §2.8 presents our refined DESC algorithm for rotation averaging, and, in particular, its initialization, which we refer to as DESC-init.

2.1. Preliminaries

The (noiseless) adversarial corruption model assumes that the set of edges E is partitioned to E_g of good (clean) edges, where $g_{ij} = g^*_{ij}$, and E_b of bad (corrupted) edges, where $g_{ij} \neq g^*_{ij}$. For each $ij \in E$, let $C_{ij} := \{k : ik, jk \in E\}$ and $G_{ij} := \{k : ik, jk \in E_g\}$. While C_{ij} and G_{ij} are sets of the nodes, we also view them as sets of cycles ijk containing edge ij, where $k \in C_{ij}$ or G_{ij}, respectively. When addressing the adversarial corruption model we further assume that for each $ij \in E$, G_{ij} is nonempty. Recall that d is a bi-invariant metric on $\mathcal{G}$ and assume WLOG that it is scaled so that $d(g_1, g_2) \in [0, 1]$ for any $g_1, g_2 \in \mathcal{G}$. Therefore, $s^*_{ij} \in [0, 1]$ for all $ij \in E$. We take advantage of the cycle inconsistency $d_{ij,k} = d(g_{ij}g_{jk}g_{ji}, e)$ and its following property (Lerman & Shi, 2021):

Proposition 2.1. *For any $ij \in E$ and $k \in C_{ij}$,*

$$|d_{ij,k} - s^*_{ij}| \leq s^*_{ik} + s^*_{jk}, \tag{4}$$

and consequently,

$$d_{ij,k} = s^*_{ij} \quad \text{if} \quad ik, jk \in E_g. \tag{5}$$

Note that (5) states that the cycle inconsistency of cycle ijk is an exact estimator of the corruption level of edge ij whenever $k \in G_{ij}$. Since we assumed that G_{ij} is nonempty for any $ij \in E$, s^*_{ij} must be supported on the set $\{d_{ij,k}\}_{k \in C_{ij}}$. The distribution of s^*_{ij} can thus be written as the probability mass for $k \in |C_{ij}|$: $\boldsymbol{p}^*_{ij}(k) = \mathbf{1}_{\{k \in G_{ij}\}}/|G_{ij}|$, where $\mathbf{1}$ is the indicator function. Indeed, by (5), $\boldsymbol{p}^*_{ij}$ only has positive mass on the k's such that $d_{ij,k} = s^*_{ij}$, so the distribution concentrates on the true value of s^*_{ij}. Let $\Delta(m)$ denote the simplex of length m, that is, $\Delta(m) = \{\boldsymbol{x} \in \mathbb{R}^m : \boldsymbol{x} \geq \mathbf{0},\ \|\boldsymbol{x}\|_1 = 1\}$, where $\boldsymbol{x} \geq \mathbf{0}$ means that all coordinates of $\boldsymbol{x}$ are nonnegative. Let $\Delta'(|C_{ij}|)$ denote the subset of $\Delta(|C_{ij}|)$ with zero coordinates whenever $k \notin G_{ij}$. Using this notation, $\boldsymbol{p}^*_{ij} \in \Delta'(|C_{ij}|) \subset \Delta(|C_{ij}|)$.

2.2. General Formulation and Motivation

Let $\boldsymbol{d}_{ij}, \boldsymbol{v}^*_{ij} \in \mathbb{R}^{|C_{ij}|}$ denote the vectors such that $\boldsymbol{d}_{ij}(k) = d_{ij,k}$ and $\boldsymbol{v}^*_{ij}(k) = s^*_{ik} + s^*_{jk}$ for $k \in C_{ij}$, respectively. We notice the following interesting relationship of $\boldsymbol{p}^*_{ij}$ and s^*_{ij}:

Proposition 2.2. *For any $ij \in E$*

$$\boldsymbol{p}^{*\top}_{ij}\boldsymbol{d}_{ij} = s^*_{ij} \quad \text{and} \quad \boldsymbol{p}^{*\top}_{ij}\boldsymbol{v}^*_{ij} = 0. \tag{6}$$

Proof. We prove the following more general result that implies (6) and we will be used later:

$$\boldsymbol{p}^\top_{ij}\boldsymbol{d}_{ij} = s^*_{ij} \text{ and } \boldsymbol{p}^\top_{ij}\boldsymbol{v}^*_{ij} = 0, \text{ for } \boldsymbol{p}_{ij} \in \Delta'(|C_{ij}|). \tag{7}$$

The definition of $\Delta'(|C_{ij}|)$ and (5) imply the first equality as follows: $\boldsymbol{p}^\top_{ij}\boldsymbol{d}_{ij} = \sum_{k \in G_{ij}} \boldsymbol{p}_{ij}(k)d_{ij,k} = s^*_{ij}$. The second equality follows from $p_{ij}(k)\boldsymbol{v}^*_{ij}(k) = p_{ij}(k)\mathbf{1}_{\{k \in G_{ij}\}}(s^*_{ik} + s^*_{jk}) = p_{ij}(k)\mathbf{1}_{\{s^*_{ik}+s^*_{jk}=0\}}(s^*_{ik} + s^*_{jk}) = 0$. Since $\boldsymbol{p}_{ij} \in \Delta'(|C_{ij}|)$, (7) implies (6). □

We remark that (6) only relies on the assumption that $|G_{ij}| > 0$, and does not assume any probabilistic model.

DESC aims to estimate both $\{\boldsymbol{p}^*_{ij}\}_{ij \in E}$ and $\{s^*_{ij}\}_{ij \in E}$, while directly using Proposition (2.2). The "detection" task in DESC refers to finding $\boldsymbol{p}^*_{ij}$, which is equivalent to detecting the set of "good" cycles ijk such that $d_{ij,k} = s^*_{ij}$. The "estimation" task in DESC refers to estimating s^*_{ij}. For $ij \in E$, let $\boldsymbol{p}_{ij} \in \mathbb{R}^{|C_{ij}|}$ and $s_{ij} \in \mathbb{R}$ denote the estimates by DESC of $\boldsymbol{p}^*_{ij}$ and s^*_{ij}. We also define $\boldsymbol{v}_{ij} \in \mathbb{R}^{|C_{ij}|}$ by $\boldsymbol{v}_{ij}(k) = s_{ik} + s_{jk}$ for $k \in C_{ij}$, so $\boldsymbol{v}_{ij}$ estimates $\boldsymbol{v}^*_{ij}$.

DESC aims to solve the following optimization problem.

$$\min_{\{s_{ij}\}_{ij \in E}, \{\boldsymbol{p}_{ij}\}_{ij \in E}} \sum_{ij \in E} \boldsymbol{p}^\top_{ij}\boldsymbol{v}_{ij} \tag{8}$$

$$\text{subject to} \quad s_{ij} = \boldsymbol{p}^\top_{ij}\boldsymbol{d}_{ij},\ ij \in E$$

$$\boldsymbol{p}_{ij} \in \Delta(|C_{ij}|),\ ij \in E.$$

Note that the constraints in (8) and the fact that d was scaled to be in $[0, 1]$ (so $d_{ij,k} \leq 1$) imply that $\boldsymbol{p}_{ij}(k)$ and s_{ij} in (8) are between 0 and 1. This formulation is clearly motivated by Proposition (2.2). Indeed, the first equation of (6) is the first constraint of DESC (the other constraint of DESC just specifies the domain of $\boldsymbol{p}_{ij}$). In order to try to enforce the second equation of (6) we aim to minimize the cumulative sum of the positive terms $\{\boldsymbol{p}^{*\top}_{ij}\boldsymbol{v}^*_{ij}\}_{ij \in E}$. When the minimum value is 0, then the second equation of (6) is satisfied for all $ij \in E$.

Another interpretation of (8) arises when bounding the cumulative estimation error of $\{s^*_{ij}\}_{ij\in E}$, using the constraints in (8) as well as (4):

$$\sum_{ij\in E}|s_{ij}-s^*_{ij}| = \sum_{ij\in E}\left|\boldsymbol{p}_{ij}^\top \boldsymbol{d}_{ij} - s^*_{ij}\right| \tag{9}$$

$$= \sum_{ij\in E}\left|\sum_{k\in C_{ij}} \boldsymbol{p}_{ij}(k)\left(\boldsymbol{d}_{ij}(k)-s^*_{ij}\right)\right|$$

$$\le \sum_{ij\in E}\sum_{k\in C_{ij}} \boldsymbol{p}_{ij}(k)\left|\boldsymbol{d}_{ij}(k)-s^*_{ij}\right|$$

$$\le \sum_{ij\in E}\sum_{k\in C_{ij}} \boldsymbol{p}_{ij}(k)(s^*_{ik}+s^*_{jk}) = \sum_{ij\in E}\boldsymbol{p}_{ij}^\top \boldsymbol{v}^*_{ij}.$$

Therefore, (8) minimizes an approximate upper bound for the cumulative error, where $\boldsymbol{v}_{ij}$ replaces $\boldsymbol{v}^*_{ij}$ in the right hand side (RHS) of (9).

2.3. DESC as a Quadratic Program

Plugging the first constraint of (8) into the objective function of (8) yields a quadratic objective function in $\{\boldsymbol{p}_{ij}\}_{ij\in E}$:

$$\sum_{ij\in E}\sum_{k\in C_{ij}} \boldsymbol{p}_{ij}(k)(s_{ik}+s_{jk})$$

$$= \sum_{ij\in E}\sum_{k\in C_{ij}} \boldsymbol{p}_{ij}(k)\left(\boldsymbol{p}_{ik}^\top \boldsymbol{d}_{ik} + \boldsymbol{p}_{jk}^\top \boldsymbol{d}_{jk}\right). \tag{10}$$

Therefore, in order to minimize (8) it is sufficient to find all minimizers of the RHS of (10) of the form $\{\widehat{\boldsymbol{p}}_{ij}\}_{ij\in E}$. The first constraint of (8) is not needed in the latter minimization, but it yields the additional minimizers of the form $\{\widehat{s}_{ij}\}_{ij\in E}$ of (8) as follows:

$$\widehat{s}_{ij} = \widehat{\boldsymbol{p}}_{ij}^\top \boldsymbol{d}_{ij} \text{ for } ij\in E. \tag{11}$$

Thus, the quadratic programming formulation of DESC is

$$\min_{\substack{\{\boldsymbol{p}_{ij}\}_{ij\in E}\\ \subset \Delta(|C_{ij}|)}} \sum_{ij\in E}\sum_{k\in C_{ij}} \boldsymbol{p}_{ij}(k)\left(\boldsymbol{p}_{ik}^\top \boldsymbol{d}_{ik} + \boldsymbol{p}_{jk}^\top \boldsymbol{d}_{jk}\right), \tag{12}$$

where in view of (11), there is a one-to-one correspondence between the minimizers of (12) and (8).

2.4. Theory for the DESC Framework

We show that under some mild conditions, the global minimum of the DESC formulation exactly recovers s^*_{ij}.

Theorem 2.3. *If $|G_{ij}|\ge 1\ \forall ij\in E$ and $\forall k\in C_{ij}\ d_{ij,k}>0$ whenever ij, jk or $ki\notin E_g$, then any global minimum of (8) exactly recovers the corruption levels $\{s^*_{ij}\}_{ij\in E}$*

Proof. Let $\{\widehat{\boldsymbol{p}}_{ij}\}_{ij\in E}$ be a minimizer of (12) and $\{\widehat{s}_{ij}\}_{ij\in E}$ be defined by (11) (so $\{\widehat{\boldsymbol{p}}_{ij}\}_{ij\in E}$, $\{\widehat{s}_{ij}\}_{ij\in E}$ is a minimizer of (8)). We will show that $\widehat{s}_{ij}=s^*_{ij}$ for all $ij\in E$ and will thus conclude the stated exact recovery.

If $\widehat{\boldsymbol{p}}_{ij}\in\Delta'(|C_{ij}|)$, then $s_{ij}=\boldsymbol{p}_{ij}^\top \boldsymbol{d}_{ij}=s^*_{ij}$, where the first equality is due to the first constraint in (8) and the second one is due to (7). Consequently, $\widehat{s}_{ij}=s^*_{ij}$ and the corruption levels are exactly recovered.

To conclude the proof we will show that $\widehat{\boldsymbol{p}}_{ij}\in\Delta'(|C_{ij}|)$. Assume on the contrary that $\widehat{\boldsymbol{p}}_{ij}(k)>0$ for some $k\notin G_{ij}$. Since $k\notin G_{ij}$, WLOG we assume that $ik\in E_b$. By our assumption on cycle-inconsistencies, we obtain that $d_{ik,l}>0$ for all $l\in C_{ik}$. Thus, $s_{ik}=\boldsymbol{p}_{ik}^\top \boldsymbol{d}_{ik}>0$ (since all elements of $\boldsymbol{d}_{ik}$ are positive and at least one element of $\boldsymbol{p}_{ik}$ is positive). Consequently, $\boldsymbol{p}_{ij}(k)s_{ik}>0$ and the value of (10) is strictly greater than 0. This contradicts the assumption that $\{\boldsymbol{p}_{ij}\}_{ij\in E}$ is a global minimum. □

We provide two immediate corollaries of Theorem 2.3.

Corollary 2.4. *Assume that $\mathcal{G}$ is a compact Lie group. If for any $ij\in E$, $|G_{ij}|\ge 1$, and for any $ij\in E_b$, g_{ij} is i.i.d. sampled from an absolutely continuous distribution over $\mathcal{G}$, then with probability 1 any global minimum of (8) exactly recovers the corruption levels $\{s^*_{ij}\}_{ij\in E}$.*

Proof. We claim that the assumptions of this corollary imply the conditions of Theorem 2.3 with probability 1. Indeed, if ijk is a 3-cycle that contains at least one edge in E_b, where WLOG this bad edge is ij, then $\Pr(d_{ij,k}=0)=\Pr(g_{ij}=g_{ik}\,g_{kj})=0$ due to the continuity of the density of g_{ij}. Thus with probability 1, $d_{ij,k}>0$. □

We remark that Corollary 2.4 does not assume a specific probabilistic distribution and thus it is more general than previous probabilistic results by Wang & Singer (2013).

Corollary 2.5. *Assume a compact Lie group $\mathcal{G}$ and data generated by UCM with n nodes, probability p of connecting two nodes p, and probability q of corrupting an edge. Then for $n/\log n\ge 10/(p^2(1-q)^2)$, with probability at least $1-n^{-0.7}$ any global minimum of (8) exactly recovers the corruption levels $\{s^*_{ij}\}_{ij\in E}$.*

Proof. It suffices to show that under the assumption of this corollary, $|G_{ij}|\ge 1$ is satisfied with high probability. We first observe that $X_k := \mathbf{1}_{\{k\in G_{ij}\}}$, for $k\in[n]$, are i.i.d. Bernoulli random variables with mean $\mu=p^2(1-q)^2$. Applying the Chernoff bound to X_k yields

$$\Pr(|G_{ij}|\ge 1)=\Pr\left(\frac{1}{n}\sum_{k\in[n]}X_k\ge\frac{\mu}{np^2(1-q)^2}\right)$$

$$>1-e^{-\frac{1}{3}\left(1-\frac{1}{np^2(1-q)^2}\right)^2 p^2(1-q)^2 n}. \tag{13}$$

If $n/\log n\ge 10/(p^2(1-q)^2)$, then $1/(np^2(1-q)^2)<1/10$ for $n>2$ and thus (13) implies that

$$\Pr(|G_{ij}|\ge 1)>1-e^{-\frac{27}{100}p^2(1-q)^2 n}.$$

By taking a union bound over $ij\in E$ and applying the assumption $n/\log n\ge c/(p^2(1-q)^2)$ for $c\ge 10$, we obtain that

$$\Pr(\min_{ij\in E}|G_{ij}|\ge 1)>1-n^2e^{-\frac{27}{100}p^2(1-q)^2 n}$$

$$\ge 1-n^2e^{-\frac{27c}{100}\log n}=1-n^{2-\frac{27c}{100}}\ge 1-n^{-0.7}.$$ □

Corollary 2.4 implies that the sample complexity for the DESC framework is $n/\log n = \Omega(p^{-2}(1-q)^{-2})$, where the order of $1-q$ matches the information-theoretic one. On the other hand, as explained in Lerman & Shi (2021), due to the use of 3-cycles one cannot improve the dependence on p.

2.5. Optimization of the DESC Framework

We optimize the quadratic program in (12) by a projected gradient descent (PGD) method. At each iteration t, let $\{s_{ij}^{(t)}\}_{ij\in E}$ and $\{\boldsymbol{p}_{ij}^{(t)}\}_{ij\in E}$ be the corresponding estimates of a minimizer of (8) (which is equivalently obtained via (12) and (11)). Denote the objective function in (12) by $f(\{\boldsymbol{p}_{ij}\}_{ij\in E})$ and its gradient with respect to $\boldsymbol{p}_{ij}$ at the estimates $p_{ij}^{(t)}$ and $s_{ij}^{(t)}$, $ij\in E$, by

$$\nabla_{ij}^{(t)} f := \left(\frac{\partial f}{\partial \boldsymbol{p}_{ij}(k)}\right)_{k\in C_{ij}} \Bigg|_{\substack{\{\boldsymbol{p}_{ij}\}_{ij\in E}=\{\boldsymbol{p}_{ij}^{(t)}\}_{ij\in E} \\ \{s_{ij}\}_{ij\in E}=\{s_{ij}^{(t)}\}_{ij\in E}}}, \quad (14)$$

where

$$\frac{\partial f}{\partial \boldsymbol{p}_{ij}(k)} = s_{ik} + s_{jk} + \left(\sum_{l\in C_{ij}} \boldsymbol{p}_{il}(j) + \boldsymbol{p}_{jl}(i)\right) d_{ij,k}. \quad (15)$$

Recall that for each $ij\in E$, $\boldsymbol{p}_{ij}\in\Delta(|C_{ij}|)$. Note that $\Delta(|C_{ij}|)$ is contained in the hyperplane

$$H_{ij} := \{\boldsymbol{x}\in\mathbb{R}^{|C_{ij}|} : \sum_{i=1}^{|C_{ij}|} x_i = 1\}$$

and that H_{ij} is the tangent space of $\Delta(|C_{ij}|)$ in $\mathbb{R}^{|C_{ij}|}$. We further note that the orthogonal projector onto H_{ij} is $\mathbf{1}\mathbf{1}^\top/|C_{ij}|$, where $\mathbf{1}$ is the all-one vector of length $|C_{ij}|$. Therefore the corresponding Riemannian gradient (Boumal, 2020) is

$$\widetilde{\nabla}_{ij}^{(t)} f = (\boldsymbol{I} - \mathbf{1}\mathbf{1}^\top/|C_{ij}|)\nabla_{ij}^{(t)} f,$$

where $\boldsymbol{I}$ denotes the identity matrix.

Our projected gradient descent method updates $\boldsymbol{p}_{ij}^{(t+1)}$ at each iteration using the Riemannian gradient and then projects onto $\Delta(|C_{ij}|)$. That is, for $ij\in E$

$$\widetilde{\boldsymbol{p}}_{ij}^{(t+1)} = \boldsymbol{p}_{ij}^{(t)} - \alpha_t \widetilde{\nabla}_{ij}^{(t)} f \quad \text{and}$$
$$\boldsymbol{p}_{ij}^{(t+1)} = \mathrm{Proj}_{\Delta(|C_{ij}|)}(\widetilde{\boldsymbol{p}}_{ij}^{(t+1)}).$$

The projector onto $\Delta(|C_{ij}|)$ can be computed following the method of Wang & Carreira-Perpinán (2013): For each fixed $ij\in E$,

$$\boldsymbol{p}_{ij}^{(t+1)} = \max(\widetilde{\boldsymbol{p}}_{ij}^{(t+1)} - \tau, 0), \quad (16)$$

where the parameter $\tau\in\mathbb{R}$ is the solution of

$$\sum_{k\in C_{ij}} \max(\widetilde{\boldsymbol{p}}_{ij}^{(t+1)}(k) - \tau, 0) = 1.$$

We remark that $h(\tau) = \sum_{k\in C_{ij}} \max(\widetilde{\boldsymbol{p}}_{ij}^{(t+1)}(k) - \tau, 0)$ is a piecewise linear function where the endpoints of the piecewise intervals are $\{\widetilde{\boldsymbol{p}}_{ij}^{(t+1)}(k)\}_k$. We order $\widetilde{\boldsymbol{p}}_{ij}^{(t+1)}(k)$ by their values from low to high, and find k such that $h(\widetilde{\boldsymbol{p}}_{ij}^{(t+1)}(k)) \le 1$ and $h(\widetilde{\boldsymbol{p}}_{ij}^{(t+1)}(k+1)) > 1$. In this way, the range of τ is narrowed down to $[\widetilde{\boldsymbol{p}}_{ij}^{(t+1)}(k), \widetilde{\boldsymbol{p}}_{ij}^{(t+1)}(k+1))$, and on this interval $h(\tau)$ is a linear function and $h(\tau)=1$ can be easily solved.

We refer to this procedure as DESC-PGD and summarize it in Algorithm 1. We remark that our proposed PGD is analogous to the Riemannian gradient descent method in Boumal (2020), except that our projection onto the simplex is not a valid retraction.

Algorithm 1 DESC-PGD

Input: $\{g_{ij}\}_{ij\in E}, \{d_{ij,k}\}_{k\in C_{ij}}, t_{\max}$
Steps:
$\boldsymbol{p}_{ij}^{(0)} = \mathbf{1}/|C_{ij}|$ $\quad ij\in E$
for $t = 1 : t_{\max}$ **do**
$\widetilde{\boldsymbol{p}}_{ij}^{(t)} = \boldsymbol{p}_{ij}^{(t-1)} - \alpha_{t-1}\widetilde{\nabla}_{ij}^{(t-1)} f$ $\quad ij\in E$
$\boldsymbol{p}_{ij}^{(t)} = \mathrm{Proj}_{\Delta(|C_{ij}|)}(\widetilde{\boldsymbol{p}}_{ij}^{(t)})$ $\quad ij\in E$
$s_{ij}^{(t)} = (\boldsymbol{p}_{ij}^{(t)})^\top \boldsymbol{d}_{ij}$ $\quad ij\in E$
end for
Output: $\widehat{s}_{ij} = s_{ij}^{t_{\max}}$

In practice, to accelerate the implementation of DESC, instead of using all the 3-cycles, one may use a randomly sampled subset. That is, for each $ij\in E$, C_{ij} is a randomly sampled set of nodes k such that $ik, jk\in E$.

2.6. Computational Complexity of DESC-PGD

At each iteration of DESC-PGD, the gradient computation in (15) requires the sum of $\boldsymbol{p}_{il}(j)$ and $\boldsymbol{p}_{jl}(i)$ over $l\in C_{ij}$ for each $ij\in E$, which takes $O(|E|c)$ computation time where c is the average of $|C_{ij}|$. The projection onto $\Delta(|C_{ij}|)$ has the same $O(|E|c)$ complexity. Since the complexity of computing $d_{ij,k}$ is also $O(|E|c)$, the per-iteration time complexity of DESC is $O(|E|c)$, which is exactly the same as that of CEMP.

2.7. Estimation of General Group Elements

We follow ideas of Lerman & Shi (2021) to estimate the group elements, $\{g_i^*\}_{i\in[n]}\subset\mathcal{G}$, using $\{\widehat{s}_{ij}\}_{ij\in E}$. We assume that $\mathcal{G}$ is a subgroup of the orthogonal group $O(D)$. For this purpose we use the graph connection weight (GCW) matrix (Singer & Wu, 2012), which aims to approximately solve the following weighted least squares problem:

$$\min_{\{g_i\}_{i\in[n]}\subset\mathcal{G}} \sum_{i\in[n]}\sum_{j\in N_i} w_{ij} d^2(g_i g_j^{-1}, g_{ij}), \quad (17)$$

where $N_i = \{j : ij\in E\}$ is a set of neighboring nodes of i and w_{ij} is a normalized graph weight such that $\sum_{j\in N_i} w_{ij} = 1$. In practice, we compute w_{ij} by normalizing $\widehat{s}_{ij}^{-3/2}$. We represent each group element, g_i, by a $D\times D$ orthogonal matrix and stack these matrices to form an $nD\times D$ block matrix $\boldsymbol{Y}$ whose i-th block is the matrix representation of g_i.

We initially estimate $\boldsymbol{Y}$ by finding the top D eigenvectors of the block matrix $\boldsymbol{X}$, where the (i,j)-th black of $\boldsymbol{X}$ is $w_{ij}g_{ij}$, and each g_{ij} is represented as its corresponding orthogonal matrix. Then we project each block of the initially estimated $\boldsymbol{Y}$ onto $\mathcal{G}$ to obtain the estimated group elements.

2.8. A Refined Solution for Rotation Averaging

For rotation averaging, we propose using the DESC-based GCW procedure of §2.7 to initialize the absolute rotations. We then suggest using the $\widehat{s}_{ij}$'s obtained by DESC-PGD to improve the IRLS algorithm of Chatterjee & Govindu (2017) and thus refine the initialized rotations.

We first briefly review the latter IRLS algorithm. For $i \in [n]$ and $t \in \mathbb{N}$, let $\boldsymbol{R}_i^{(t)}$ denote the absolute rotation matrix estimated by IRLS at iteration t. For $ij \in E$, let $\boldsymbol{R}_{ij}$ denote the input relative rotation matrix. IRLS updates at each iteration the estimated absolute rotations. Given $\boldsymbol{R}_i^{(t-1)}$, $i \in [n]$, it solves an optimization problem for the matrices $\Delta\boldsymbol{R}_i^{(t)}$, $i \in [n]$, which satisfy $\boldsymbol{R}_i^{(t)} = \boldsymbol{R}_i^{(t-1)}\Delta\boldsymbol{R}_i^{(t)}$ (note that $\Delta\boldsymbol{R}_i^{(t)}$ approaches $\boldsymbol{I}$ as t approaches infinity). The desired optimization is the weighted least squares of (17) with iteratively updated edge weights, $w_{ij}^{(t)}$, $ij \in E$, where g_i, g_j, g_{ij} are replaced by $\Delta\boldsymbol{R}_i^{(t)}$, $\Delta\boldsymbol{R}_j^{(t)}$, $(\boldsymbol{R}_i^{(t-1)})^\top\boldsymbol{R}_{ij}\boldsymbol{R}_j^{(t-1)}$, respectively. This formulation is further approximated by mapping, at each iteration, the rotation matrices in $SO(3)$ to the tangent space of $\boldsymbol{I}$, $\mathfrak{so}(3)$, by the matrix logarithm, log. We denote the mapped elements by $\Delta\Omega_i^{(t)} = \log\Delta\boldsymbol{R}_i^{(t)}$, $i \in [n]$, and

$$\Delta\Omega_{ij}^{(t)} = \log((\boldsymbol{R}_i^{(t-1)})^\top\boldsymbol{R}_{ij}\boldsymbol{R}_j^{(t-1)}), \quad ij \in E. \tag{18}$$

Chatterjee & Govindu (2017) approximate (17) by minimizing over $\{\Delta\boldsymbol{\Omega}_i^{(t)}\}_{i\in[n]} \subset \mathfrak{so}(3)$ the function

$$\sum_{ij\in E} w_{ij}^{(t)}\|\Delta\boldsymbol{\Omega}_i^{(t)} - \Delta\boldsymbol{\Omega}_j^{(t)} - \Delta\boldsymbol{\Omega}_{ij}^{(t)}\|_F^2, \tag{19}$$

Next, they compute for any edge $ij \in E$ the residual $r_{ij}^{(t)} := \|\Delta\boldsymbol{\Omega}_i^{(t)} - \Delta\boldsymbol{\Omega}_j^{(t)} - \Delta\boldsymbol{\Omega}_{ij}^{(t)}\|_F/\sqrt{2\pi^2}$ and update the weights by $w_{ij}^{(t+1)} = (r_{ij}^{(t)})^{-3/2}$. The basic idea is that edges with higher residuals are likelier to be corrupted and thus should be assigned smaller weights.

We modify this IRLS procedure as follows. First, we initialize the rotations by GCW, which uses the output of DESC-PGD (see §2.7). Our numerical experiments indicate that this initialization is often more accurate than IRLS (Chatterjee & Govindu, 2017). Second, we replace the residuals $r_{ij}^{(t)}$ in IRLS by a convex combination of $r_{ij}^{(t)}$ and DESC-estimated $\widehat{s}_{ij}$, where the coefficient of $r_{ij}^{(t)}$ is $t/(t+1)$. Consequently, the information from the residual is increasingly emphasized and $\widehat{s}_{ij}$ is mainly used to guide IRLS to escape the local minima in the first few iterations. At last, after computing the edge weights, we assign the weight 10^{-8} to a certain percentage of the edges with the lowest weights (the chosen percentage at iteration t is $\min(5t\,, 20)$). We do not assign 0 weights (i.e., completely remove them) in order to avoid a disconnected graph. The last two ideas are also used in MPLS (Shi & Lerman, 2020)). Nevertheless, our rotation refinement is also different from MPLS in the following ways. First, MPLS uses a minimal spanning tree to initialize rotations which results in inaccuracies when all edges are noisy. Second, MPLS also uses a message passing unit to update edge weights in each iteration, which is more complex than our method.

Algorithm 2 describes our overall solution to rotation averaging, which we refer to as DESC-$SO(3)$, or just DESC. We refer to the initialization of this solution (obtained in the second step of Algorithm 2) by DESC-init.

Algorithm 2 DESC-$SO(3)$ (DESC)

Input: $\{\boldsymbol{R}_{ij}\}_{ij\in E}, \{d_{ij,k}\}_{k\in C_{ij}}$
Steps:
Compute $\{\widehat{s}_{ij}\}_{ij\in E}$ by DESC-PGD
Initialize $\{\boldsymbol{R}_i^0\}_{i\in[n]}$ by DESC-based GCW (see §2.7).
$t = 0$
$w_{ij}^{(0)} = \min(\widehat{s}_{ij}^{-3/2}, 10^8)$ $\quad ij \in E$
while not convergent **do**
$t = t + 1$
Compute $\Delta\boldsymbol{\Omega}_{ij}^{(t)}$ according to (18) $\quad ij \in E$
Find $\{\Delta\boldsymbol{\Omega}_i^{(t)}\}_{i\in[n]}$ as the minimizer of (19) over $\mathfrak{so}(3)^n$
$\boldsymbol{R}_i^{(t)} = \boldsymbol{R}_i^{(t-1)}\exp(\Delta\boldsymbol{\Omega}_i^{(t)})$ $\quad i \in [n]$
$r_{ij}^{(t)} = \|\Delta\boldsymbol{\Omega}_i^{(t)} - \Delta\boldsymbol{\Omega}_j^{(t)} - \Delta\boldsymbol{\Omega}_{ij}^{(t)}\|_F/(\sqrt{2}\pi)$ $\quad ij \in E$
$h_{ij}^{(t)} = (t \cdot r_{ij}^{(t)} + \widehat{s}_{ij})/(t+1)$ $\quad ij \in E$
$w_{ij}^{(t)} = \min((h_{ij}^{(t)})^{-3/2}, 10^8)$ $\quad ij \in E$
$\tau_t = \min(5t\,, 20)$
$w_{ij}^{(t)} = 10^{-8}$ for $\tau_t\%$ of edges with the highest $h_{ij}^{(t)}$
end while
Output: $\left\{\boldsymbol{R}_i^{(t)}\right\}_{i\in[n]}$

3. Experiments

We test our methods for rotation averaging. In §3.1, we describe the implementation details of all tested algorithms. In §3.2 we report the estimation of the corruption levels and rotations on synthetic data generated by UCM for $SO(3)$. In §3.3, we compare the performance of different algorithms on the Photo Tourism dataset (Wilson & Snavely, 2014).

3.1. Implementation Details of All Algorithms

We first compare our QP scheme with the following linear programming (LP) method for estimating corruption levels:

$$\min_{s_{ij}} \sum_{ij\in E} s_{ij} \tag{20}$$
$$\text{subject to } |s_{ij} - d_{ij,k}| \le s_{ik} + s_{jk}$$
$$0 \le s_{ij} \le 1,$$

where the first constraint is due to (4). We solve (20) using the default Matlab CVX LP solver. Using this solution for the corruption levels, one can apply the same post-processing as DESC to estimate the rotations (see §2.7). This LP formulation is very similar to that of Agarwal et al. (2020) except that Agarwal et al. (2020) is designed for rank aggregation. It is also similar to that of Zach et al. (2010), but Zach et al. (2010) with an additional penalty in the form of the sum over cycles of maximal corruption levels within each cycle. The objective function in (20) is based on the assumption that the overall corruption level of the graph is small and thus does not apply to highly corrupted scenarios. In contrast, DESC aims to enforce the orthogonality of $\boldsymbol{v}_{ij}$ and $\boldsymbol{p}_{ij}$ (see (8)), which seems also relevant to high corruption.

We also compare DESC with DESC-init and competitive GS methods. We test two versions of IRLS: IRLS-GM (Chatterjee & Govindu, 2013) and IRLS-$\ell_{1/2}$ (Chatterjee & Govindu, 2017), while using their default implementations. These versions use the Geman McClure (GM) and $\ell_{1/2}$ losses. We implement CEMP (Lerman & Shi, 2021) and MPLS (Shi & Lerman, 2020) using the codes provided by the respective papers, with their default parameters. Following Lerman & Shi (2021), we use CEMP for recovering the corruption levels and both CEMP+MST and CEMP+GCW to recover rotations. CEMP+MST uses the minimum spanning tree (MST) as a post-processing step to estimate rotations, and CEMP+GCW uses GCW as in §2.7.

For the synthetic data experiments, we ran DESC with a constant step size of 0.01. The maximum number of iterations was set to 100. We noticed that increasing this number improved the accuracy, but we preferred a reasonable runtime (we will discuss the tradeoff between the two later). To further reduce the runtime, we also sampled (without replacement) a subset of the cycles of each edge. The number of cycles sampled was chosen as one quarter of the median number of cycles per edge, or at least 30. For edges with fewer cycles than the sample number, all cycles were used. No other parameters needed to be tuned.

For real data, due to the large sizes of the datasets, we increased the step size to 1 in order to accelerate the convergence and we decreased the maximum number of iterations to 30. Otherwise, all parameter settings were identical.

3.2. Synthetic Data Experiments

We compare DESC and other algorithms on synthetic data generated according to UCM, with and without noise. The underlying graph is generated by an Erős-Rényi model $G(n,p)$ where $n = 100$ and $p = 0.5$ (two nodes are connected by an edge with probability p). The group is $\mathcal{G} = SO(3)$ and we represent its elements (rotations) by 3×3 rotation matrices. Let $\mathrm{Haar}(SO(3))$ denote the Haar (or "uniform") probability measure on $SO(3)$ and for $ij \in E$, let W_{ij} be a 3×3 Wigner matrix (with i.i.d. standard normal elements). For $0 \leq q < 1$ and $\sigma \geq 0$, the following corruption model generates the rotation measurements:

$$g_{ij} = \begin{cases} \mathrm{Proj}(g^*_{ij} + \sigma W_{ij}), & \text{with probability } 1-q; \\ \tilde{g}_{ij} \sim \mathrm{Haar}(SO(3)), & \text{with probability } q. \end{cases}$$

That is, a group element is corrupted with probability q and in this case it is i.i.d. sampled from the "uniform" measure on $SO(3)$ and otherwise its value is obtained by adding noise to the the ground-truth group ratio g^*_{ij} with constant noise level σ. The resulting noisy matrix is then projected to $SO(3)$.

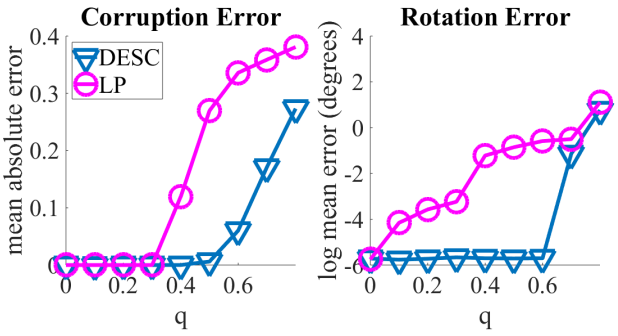

Figure 1. Left: mean absolute error for corruption estimation of both DESC and linear programming, Right: log mean error in degrees for the rotation estimates of DESC and linear programming.

Using synthetic data generated from this model, we first compare our QP formulation with the LP formulation of §3.1. Since both of them aim to find the corruption levels, we compute the following absolute error for corruption estimation:

$$\frac{1}{|E|} \sum_{ij \in E} |\widehat{s}_{ij} - s^*_{ij}|. \tag{21}$$

Figure 1 shows the absolute mean errors for corruption estimation and log mean errors for rotation estimation of both LP and QP with $\sigma = 0$ and varying q. In its first plot (on left), QP performs significantly better than LP for corruption estimation when $q \geq 0.4$. This is due to the underlying assumption of the LP formulation that the overall edge corruption level is small. In its second plot (on right), DESC significantly outperforms LP with all values of $q > 0$. For fair comparison, we post-processed LP for rotation estimation with the same steps of Algorithm 2. Interestingly, even when q is small, the rotation estimates of LP are much worse than DESC, unlike the corruption estimates. The reason is that LP tends to underestimate the corruption levels due to its objective function. Underestimation of a small fraction of corruption levels as nearly 0 results in nearly infinite edge weights of the corresponding edges and consequently inaccurate rotation estimation. Due to the overall poor performance of LP, we ignore it in the rest of the experiments.

Next, we ran all algorithms, except LP, on synthetic datasets generated with $q = 0, 0.1, 0.2, \ldots, 0.8$ and both $\sigma = 0$

Table 1. Average of the mean and median errors (in degrees) for rotation estimates across the 13 datasets of Photo Tourism

	DESC	DESC-init	IRLS-GM	IRLS-$L_{\frac{1}{2}}$	CEMP-MST	CEMP-GCW	MPLS
mean	**3.5119**	3.8354	3.9644	3.8447	4.1447	3.9191	3.7142
median	**1.5938**	1.8516	1.7255	1.7201	1.7975	2.0339	1.7032

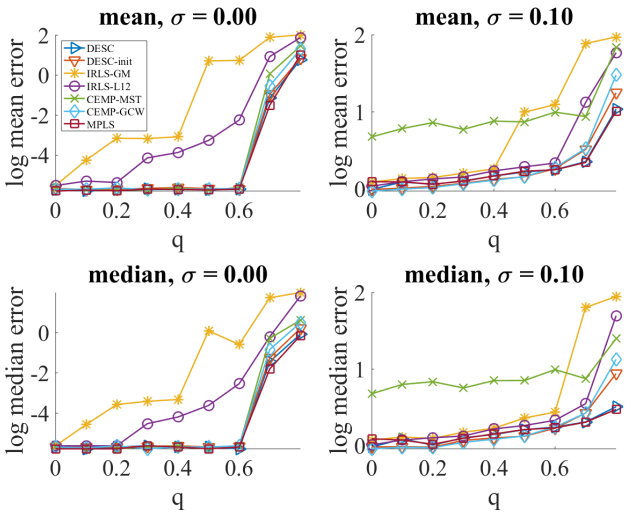

Figure 2. Mean and median errors (in degrees) for rotation estimation of different algorithms (see legend) using the synthetic data with varying q and σ. Top: mean, bottom: median, left: $\sigma = 0$ and right $\sigma = 0.1$. We applied log base 10 to the y axis.

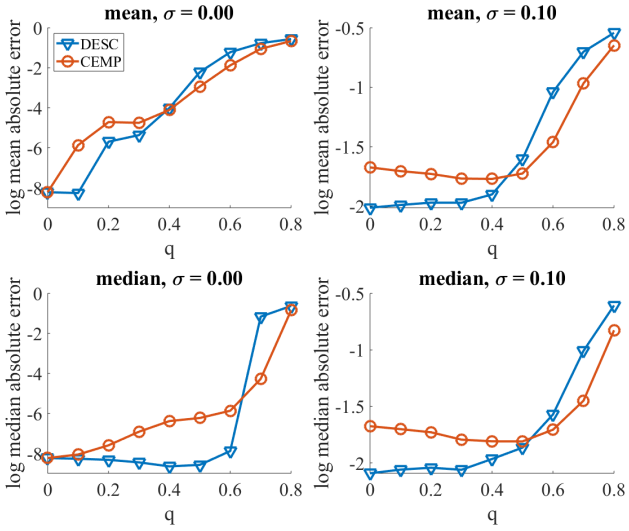

Figure 3. Mean and median absolute error of corruption estimation for DESC and CEMP using the synthetic data. The upper two plots show the means and the lower two plots show the medians. The y axis uses a logarithmic scale with base 10.

and $\sigma = 0.1$. Figure 2 reports the mean and median errors of rotation estimates by all tested methods. Because the values varied by several orders of magnitude, we used a logarithmic scale (base 10) for the y-axis. In all cases, DESC is comparable to MPLS. We note that DESC-init consistently outperforms CEMP-GCW, where both methods use the same GCW postprocessing for rotation estimation.

Figure 3 shows the absolute estimation errors of the corruption levels (see (21)) by DESC and CEMP using a logarithmic y-axis scale as in Figure 2. We note that overall the accuracy of DESC and CEMP for corruption estimation is comparable. In particular, in terms of the mean estimation error, DESC is more successful when q is small, whereas CEMP is more advantageous when q is large. In terms of the median error, DESC consistently outperforms CEMP for almost all values of q when $\sigma = 0$ and is several orders of magnitudes more accurate. When $\sigma = 0.1$, DESC yields slightly higher median error than that of CEMP for high q, and has much lower median error than CEMP for low q.

The per-iteration runtimes of DESC and CEMP on synthetic data were 0.06 and 0.02 seconds, respectively. While DESC is fast per iteration, it requires dozens of iterations to converge, making it slower than CEMP, even though both of them have the same order of computational complexity.

3.3. Real Data Experiments

For experiments with real data, we used the Photo Tourism dataset, which was introduced in Wilson & Snavely (2014). It contains hundreds of images along with the approximate ground truth rotations estimated by the bundler software (Snavely et al., 2006). The relative rotations are estimated following the pipeline presented in Ozyesil & Singer (2015). We ran DESC along with the above benchmarks (excluding LP) on the 14 Photo Tourism datasets.

Figures 4 and 5 report the mean and median rotation errors, respectively, in degrees. The Gendarmenmarkt dataset is not included because all methods performed very poorly on it, with over 30 degrees error, which skewed the scale of the y axis. We note that DESC is overall competitive. The performance of all methods widely vary, though they are fairly consistent with each other for most datasets. Table 1 shows the average of the mean and median errors of all methods across all datasets. DESC performs the best by both metrics.

Next, we tested the ability of DESC to estimate edge corruptions. Figures 6 and 7 report the mean and median error of

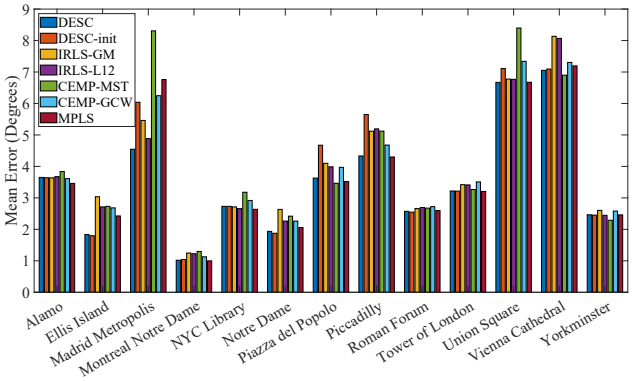

Figure 4. Mean error (in degrees) of rotation estimation for each algorithm on 13 of the Photo Tourism datasets.

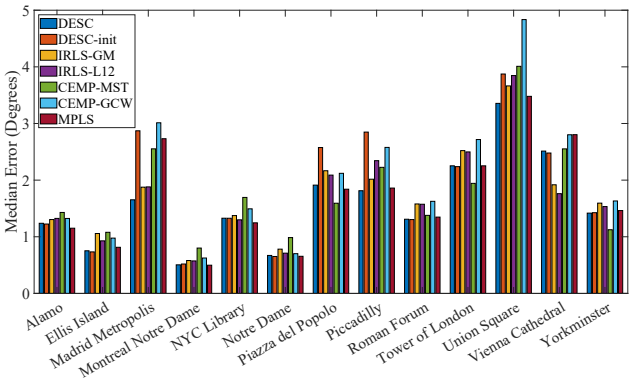

Figure 5. Median error (in degrees) of rotation estimation for each algorithm on 13 of the Photo Tourism datasets.

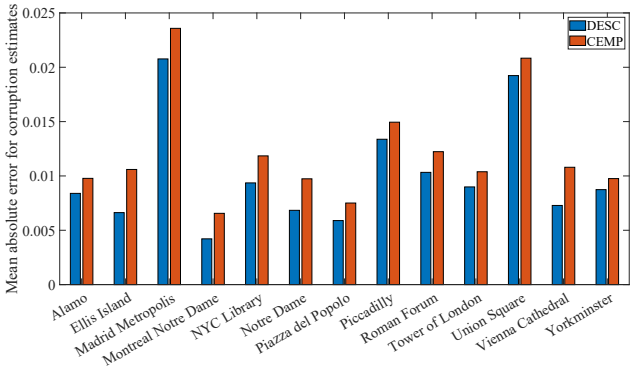

Figure 6. Mean absolute error for the corruption estimates of DESC and CEMP on 13 of the Photo Tourism datasets.

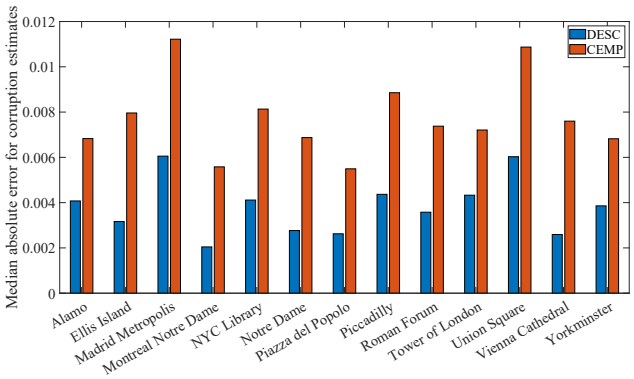

Figure 7. Median absolute error for the corruption estimates of DESC and CEMP on 13 of the Photo Tourism datasets.

corruption estimation, respectively, of both DESC and CEMP. Clearly, DESC is more successful than CEMP in recovering the corruption levels. In particular, the median error of DESC is more than 50% lower than that of CEMP on six datasets.

4. Conclusion

We proposed DESC, a novel framework for estimating the corruption levels of group ratios in group synchronization. It has a clear interpretation and we proved its exact recovery under a mild generic condition. We also established a tight recovery bound in terms of the corruption parameter under UCM. We proposed a simple numerical strategy that aimed to solve the optimization problem of DESC. We explained how to use it to solve the underlying group elements. We further refined this solution for the special case of rotation averaging. Our experiments on synthetic and real data of rotation averaging indicated that our proposed method often outperforms CEMP in corruption estimation and is competitive with state-of-the-art algorithms for rotation averaging.

Nevertheless, our method also has some limitations. First, our gradient descent algorithm is typically slower than CEMP. Second, when initializing the group elements, our edge weights are updated by $\widehat{s}_{ij}^{-3/2}$, which is quite heuristic. Consequently, an improvement in corruption estimation may not always result in improvement in rotation estimation.

In the future, we would like to study faster algorithms for optimizing our DESC formulation, and optimal ways of assigning edge weights under certain probabilistic models. We also plan to extend the idea behind our DESC framework to other tasks with structural consistency, such as subspace recovery and rank aggregation. We can also generalize our method to incorporate longer cycles in order to handle sparse graphs. For better numerical efficiency, we can use the ideas of Guibas et al. (2019) for sampling a smaller set of clean cycles.

Acknowledgement

This work was supported by NSF awards 1821266, 2124913.

References

Agarwal, A., Agarwal, S., Khanna, S., and Patil, P. Rank aggregation from pairwise comparisons in the presence of adversarial corruptions. In *International Conference on Machine Learning*, pp. 85–95. PMLR, 2020.

Bandeira, A. S. Random Laplacian matrices and convex relaxations. *Foundations of Computational Mathematics*, 18(2):345–379, 2018. doi: 10.1007/s10208-016-9341-9.

Birdal, T., Simsekli, U., Eken, M. O., and Ilic, S. Bayesian pose graph optimization via Bingham distributions and tempered geodesic MCMC. *Advances in Neural Information Processing Systems*, 31, 2018.

Birdal, T., Arbel, M., Simsekli, U., and Guibas, L. J. Synchronizing probability measures on rotations via optimal transport. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1569–1579, 2020.

Boumal, N. An introduction to optimization on smooth manifolds. *Available online, Aug*, 2020.

Chatterjee, A. and Govindu, V. M. Efficient and robust large-scale rotation averaging. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pp. 521–528, 2013.

Chatterjee, A. and Govindu, V. M. Robust relative rotation averaging. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):958–972, 2017.

Chen, Y., Guibas, L. J., and Huang, Q. Near-optimal joint object matching via convex relaxation. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 100–108, 2014.

Chen, Y., Suh, C., and Goldsmith, A. J. Information recovery from pairwise measurements. *IEEE Trans. Inf. Theory*, 62(10):5881–5905, 2016.

Cucuringu, M. Synchronization over Z2 and community detection in signed multiplex networks with constraints. *J. Complex Networks*, 3(3):469–506, 2015.

Eriksson, A., Olsson, C., Kahl, F., and Chin, T.-J. Rotation averaging and strong duality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 127–135, 2018.

Govindu, V. M. Lie-algebraic averaging for globally consistent motion estimation. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), 27 June - 2 July 2004, Washington, DC, USA*, pp. 684–691, 2004.

Guibas, L. J., Huang, Q., and Liang, Z. A condition number for joint optimization of cycle-consistent networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Hartley, R. I., Aftab, K., and Trumpf, J. L1 rotation averaging using the weiszfeld algorithm. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pp. 3041–3048, 2011.

Lerman, G. and Shi, Y. Robust group synchronization via cycle-edge message passing. *Foundations of Computational Mathematics*, pp. 1–77, 2021.

Ling, S. Near-optimal performance bounds for orthogonal and permutation group synchronization via spectral methods. *arXiv preprint arXiv:2008.05341*, 2020.

Maunu, T. and Lerman, G. Depth descent synchronization in $\mathrm{SO}(d)$, 2020. URL https://arxiv.org/abs/2002.05299.

Ozyesil, O. and Singer, A. Robust camera location estimation by convex programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2674–2683, 2015.

Pachauri, D., Kondor, R., and Singh, V. Solving the multiway matching problem by permutation synchronization. In *Advances in Neural Information Processing Systems 26*, pp. 1860–1868. Curran Associates, Inc., 2013.

Shen, T., Zhu, S., Fang, T., Zhang, R., and Quan, L. Graph-based consistent matching for structure-from-motion. In *European Conference on Computer Vision*, pp. 139–155. Springer, 2016.

Shi, Y. and Lerman, G. Message passing least squares framework and its application to rotation synchronization. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Singer, A. Angular synchronization by eigenvectors and semidefinite programming. *Applied and computational harmonic analysis*, 30(1):20–36, 2011.

Singer, A. and Wu, H.-T. Vector diffusion maps and the connection Laplacian. *Comm. Pure Appl. Math.*, 65(8):1067–1144, 2012. ISSN 0010-3640. doi: 10.1002/cpa.21395.

Snavely, N., Seitz, S. M., and Szeliski, R. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*, pp. 835–846, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-364-6.

Sun, Y., Zhuo, J., Mohan, A., and Huang, Q. K-best transformation synchronization. In *Proceedings of the*

IEEE/CVF International Conference on Computer Vision, pp. 10252–10261, 2019.

Wang, L. and Singer, A. Exact and stable recovery of rotations for robust synchronization. *Information and Inference*, 2013.

Wang, W. and Carreira-Perpinán, M. A. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*, 2013.

Wilson, K. and Snavely, N. Robust global translations with 1dsfm. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*, pp. 61–75, 2014.

Zach, C., Klopschitz, M., and Pollefeys, M. Disambiguating visual relations using loop constraints. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pp. 1426–1433, 2010.