2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) | 978-1-6654-6946-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CVPR52688.2022.01528

Fast, Accurate and Memory-Efficient Partial Permutation Synchronization

Shaohan Li[*] Yunpeng Shi[†] Gilad Lerman[*]
[*]School of Mathematics, University of Minnesota
[†]Program in Applied and Computational Mathematics, Princeton University
{li000743, lerman}@umn.edu, yunpengs@princeton.edu

Abstract

Previous partial permutation synchronization (PPS) algorithms, which are commonly used for multi-object matching, often involve computation-intensive and memory-demanding matrix operations. These operations become intractable for large scale structure-from-motion datasets. For pure permutation synchronization, the recent Cycle-Edge Message Passing (CEMP) framework suggests a memory-efficient and fast solution. Here we overcome the restriction of CEMP to compact groups and propose an improved algorithm, CEMP-Partial, for estimating the corruption levels of the observed partial permutations. It allows us to subsequently implement a nonconvex weighted projected power method without the need of spectral initialization. The resulting new PPS algorithm, MatchFAME (Fast, Accurate and Memory-Efficient Matching), only involves sparse matrix operations, and thus enjoys lower time and space complexities in comparison to previous PPS algorithms. We prove that under adversarial corruption, though without additive noise and with certain assumptions, CEMP-Partial is able to exactly classify corrupted and clean partial permutations. We demonstrate the state-of-the-art accuracy, speed and memory efficiency of our method on both synthetic and real datasets.

1. Introduction

The problem of partial permutation synchronization (PPS) naturally arises from the task of multi-object matching (MOM). MOM assumes multiple objects (e.g. images), where each single object contains some keypoints associated with underlying distinct labels. The set of all distinct labels is called the universe. Ideally, any two keypoints, from different objects, that share a common label should be matched. Given partially observed and corrupted pairwise keypoint matches, MOM asks to recover the ground truth labels of each keypoint, or equivalently, the keypoint-to-universe matches. In structure from motion (SfM), where the objects are images, MOM is often referred to as multi-image matching. Here a keypoint is characterized by a specific location in the image and its associated label is the index of its corresponding 3D point, which can be viewed at this location of the image. In this case, the initial pairwise keypoint matches are typically obtained by SIFT [12] and the MOM problem asks to identify the corresponding 3D point index for each keypoint in each image, up to an arbitrary permutation of the indices.

The mathematical formulation of PPS represents the images in the latter problem as nodes of an unweighted and undirected graph, which is commonly referred to as the viewing graph, and further represents both "relative" keypoint-to-keypoint matches among pairs of images and the "absolute" keypoint-to-universe matches as partial permutation matrices. We recall that a partial permutation matrix is binary with at most one nonzero element at each row and column. PPS thus asks to recover the "absolute" partial permutations (which are associated with nodes of the graph) given possibly corrupted and noisy measurements of the "relative" partial permutations (which are associated with edges of the graph). We remark that when restricting the partial permutation matrices to be full permutations (bi-stochastic, binary and square), PPS reduces to permutation synchronization (PS), which is a special case of the group synchronization problem. Probably, the most well-known group synchronization problem is rotation averaging [4,8,19] where the group is $SO(3)$. Various approaches from rotation averaging, such as the spectral method [22] and SDP relaxation [24], can be similarly employed to PS [9, 16]. However, when considering PS for image matching, all images must share the same set of keypoints. This is restrictive, since images taken from different viewing directions may share very few (or even no) keypoints. Therefore, PPS is more realistic for image matching, and in particular, SfM, than PS.

The PPS problem is challenging for three different reasons. First of all, the corruption of pairwise measurements in real data can be highly nonuniform, which violates the common assumptions of uniform corruption in [6, 16]. Indeed, as is pointed out in [20], the corruption in keypoint matches in real data can concentrate at local regions of the viewing graph. Second, the number of rows or columns of each partial permutation can be in the order of hundreds or even higher, which is much larger than dimension 3 in rotation averaging. This makes PPS a computationally-intensive task in comparison to rotation averaging. Finally, the relative partial permutations in PPS are no longer square matrices, as in PS, and can have very different

Authorized licensed use limited to: University Of Minnesota Duluth. Downloaded on August 30,2023 at 03:01:47 UTC from IEEE Xplore. Restrictions apply.

sizes and sparsity levels. Consequently, they may introduce additional bias and numerical instability to common PPS algorithms.

This work addresses the above challenges and develops a fast, accurate and memory efficient PPS algorithm that works well for nontrivial corruption models and large scale real data.

1.1. Related Works

The first PS algorithm [16], which is commonly referred to as Spectral, computes the top m eigenvectors of the block matrix of relative permutations, where m is the universe size. It then obtains the absolute partial permutations by projecting the blocks of the eigenmatrix to full permutations using the Hungarian algorithm [14]. It can be easily adapted to PPS tasks by using other heuristic projection methods [2, 25]. A similar PPS algorithm is MatchEig [13], which applies a faster heuristic projection to partial permutations, and an additional hard thresholding step. Although it achieves slight speedup in comparison to Spectral, the hard thresholding can result in overly sparse keypoint matches on some datasets. A theoretically guaranteed SDP relaxation method for PPS, MatchLift [6], was proposed for near-optimal handling of the uniform corruption model. However, the SDP relaxation suffers from high computational complexity and is often several orders of magnitude slower than spectral-based methods. MatchALS [28] replaces the SDP constraint of MatchLift by linear ones, which yield significant speedup. However, it is still much slower than spectral-based methods and is not scalable to even medium-size datasets. Moreover, the accuracy of [6, 28] are not competitive on some real datasets as reported in [13]. For PS, [3] relaxes the space of permutations to the Birkhoff polytope and solves the maximum a-posteriori (MAP) problem on this relaxed manifold. This method relies on a special probabilistic model for permutations, but it has no convergence guarantees and is restricted to PS.

Most importantly, all the aforementioned PS/PPS methods are memory demanding and thus cannot handle large scale SfM datasets such as Photo Tourism [23]. For spectral and MatchEig, the top m eigenvectors form a dense $M\times m$ matrix. For Photo Tourism, $m > 10^4$ and $M \approx 10^6$ and thus Spectral and MatchEig require at least 80 GB memory and cannot be implemented on a personal computer. MatchALS and MatchLift further require eigenvalue computation of an $M \times M$ dense matrix. These dense matrix operations also increase the time complexity.

A faster and more memory-efficient algorithm (excluding its initialization stage) is the projected power method (PPM). PPM is a nonconvex method based on blockwise power iterations followed by a projection onto the permutation matrices [5]. It can be equivalently viewed as a special case of the projected block coordinate descent algorithm assuming the least squares objective function. It was applied for PS, but as we show it can be easily extended to PPS. We note that in PPS each power iteration only stores $M\times m$ sparse binary matrices for estimating the absolute partial permutations. The number of nonzero elements in these sparse matrices is at most m, thus PPM is at least 10,000 times more memory-efficient than spectral and MatchEig on large SfM data. Moreover, since the above power iterations only operate on sparse matrices, its time complexity is also significantly smaller than those of spectral-based methods. However, as far as we know, PPM was only tested in [5, 10, 20] on both $\mathbb{Z}_m$-synchronization and PS, and was never applied to the PPS problem. We remark that there are a couple of limitations that prevent the application of PPM to large-scale real SfM datasets. First of all, as a nonconvex method, it is very sensitive to the initialization of the absolute permutations. A common initializer for PPM is Spectral, which makes PPM memory demanding, regardless how memory-efficient the power iteration is, and also slower. Second, same as Spectral and SDP methods, PPM minimizes the least squares energy, making it nonrobust under nonuniform and adversarial corruption as shown in [20].

The recent theoretically-guaranteed cycle-edge message passing (CEMP) algorithm [11] opens the door for fast, memory efficient, and outlier-robust implementation for compact group synchronization without spectral initialization. Different from the previous cycle-consistency-based methods [1, 7, 18, 27], it uses a fast iterative message passing scheme to globally estimate the corruption levels of the given pairwise measurements. It is numerically demonstrated in [11] that CEMP is memory-efficient and fast for $SO(d)$-synchronization, especially for large d. For permutation synchronization, [20] proposed an efficient implementation of CEMP. In particular, it iteratively reweighted the edges of the viewing graph using CEMP-estimated corruption levels, and simultaneously applied a weighted spectral/PPM method. However, these frameworks were only fully developed for compact group synchronization. The extension of CEMP to partial permutations is nontrivial, since partial permutations do not form a group and the theory of CEMP to date no longer holds in this new regime.

FCC [21], which was published after the submission of this work, assigns for each keypoint match a confidence score for being a correct match. It is fast, accurate and memory efficient, and seems to comparably perform to the proposed method. Its different graph model might be more tolerant to additive noise. However, our method has several advantages. First, it enjoys some theoretical guarantees. Second, its refined matches are automatically cycle-consistent. Last, its space complexity is lower than that of FCC with default parameters.

1.2. Contributions of This Work

The main contributions of this work are as follows:

- We overcome the restriction of CEMP to compact groups, and extend it to PPS. The new CEMP-Partial algorithm can be applied to general SfM matching data, and is theoretically guaranteed under adversarial corruption.
- We propose MatchFAME (Fast, Accurate and Memory-Efficient Matching) for PPS. It combines CEMP-Partial with weighted PPM. It only involves sparse matrix operations and thus enjoys significantly lower time and

Authorized licensed use limited to: University Of Minnesota Duluth. Downloaded on August 30,2023 at 03:01:47 UTC from IEEE Xplore. Restrictions apply.

space complexities than previous PPS methods.

- We demonstrate the accuracy and efficiency of MatchFAME on synthetic and real datasets in comparison to the current state-of-the-art PPS methods.

2. Partial Permutation Synchronization

Recall that a partial permutation matrix is a binary matrix that has at most one nonzero element at each row and column. We note that it is different from a full permutation whose rows and columns have exactly one nonzero element. A partial permutation can be rectangular, whereas the full one has to be a square. We denote by $\mathcal{P}^{l_1,l_2}$ the space of $l_1 \times l_2$ partial permutations, and by $\mathcal{P}_F^m$ the space of $m \times m$ full permutations. We also denote $[n] = \{1,2,...,n\}$ for $n \in \mathbb{N}$. Using the above notation, we formally state the PPS problem as follows. The problem assumes a graph $G = ([n],E)$ with n nodes and underlying unknown ground-truth partial permutations $\{\boldsymbol{P}_i^*\}_{i\in[n]}$ of size $m_i \times m$ associated with the nodes, where m is fixed and $m \geq m_i$ for all i. It further assumes that for each edge $ij \in E$, a pairwise partial permutation $\boldsymbol{X}_{ij}$ is observed, which is viewed as a measurement of the ground-truth partial permutation, $\boldsymbol{X}_{ij}^* := \boldsymbol{P}_i^* \boldsymbol{P}_j^{*T}$. The PPS problem asks to recover $\{\boldsymbol{P}_i^*\}_{i\in[n]}$ from $\{\boldsymbol{X}_{ij}\}_{ij\in E}$. In practice, PPS algorithms often operate on the block matrix $\boldsymbol{X} = (\boldsymbol{X}_{ij})_{1\leq i,j\leq n}$, where $\boldsymbol{X}_{ij} = 0$ for $ij \notin E$.

The adversarial corruption model partitions the edge set E into a set of clean (good) edges, E_g, and a set of corrupted (bad) edges, E_b, where for $ij \in E_g$, $\boldsymbol{X}_{ij} = \boldsymbol{X}_{ij}^*$, and for $ij \in E_b$, $\boldsymbol{X}_{ij} \neq \boldsymbol{X}_{ij}^*$. It is adversarial since it does not make assumptions on the distribution of the corrupted partial permutations and the graph topology; though we may add some assumptions.

In multi-image matching, $G([n],E)$ is referred to as the viewing graph, m is the number of 3D points and n is the number of images. Each graph node i is associated with an image with m_i keypoints, and $\boldsymbol{P}_i^*$ encodes its ground truth keypoint-universe matches. Specifically, $\boldsymbol{P}_i^*(k,l) = 1$ if and only if the k-th keypoint in image i corresponds to the l-th point in the 3D point cloud. For each edge $ij \in E$, the partial permutation $\boldsymbol{X}_{ij}$ represents the observed keypoint matches between images i and j (obtained by e.g., SIFT). We note that $\boldsymbol{X}_{ij}(k,l) = 1$ if and only if we observe a match between the k-th keypoint in image i and the l-th keypoint in the j-th image. We denote $M := \sum_{i=1}^n m_i$ and note that the block matrix $\boldsymbol{X}$ is of size $M \times M$. At last we comment that in multi-image matching one mainly cares about improving the keypoint matches. Therefore instead of the estimates $\{\hat{\boldsymbol{P}}_i\}_{i\in[n]}$ of absolute permutations, it is common to output the estimates of relative permutations, $\boldsymbol{Z}_{ij} = \hat{\boldsymbol{P}}_i \hat{\boldsymbol{P}}_j^\top$ for any $ij \in E$.

3. Proposed Method

3.1. Brief Review of CEMP

We focus on the case of PS with the distance

$$d(\boldsymbol{X}_1,\boldsymbol{X}_2) = 1 - \langle \boldsymbol{X}_1,\boldsymbol{X}_2\rangle/m, \text{ for } \boldsymbol{X}_1, \boldsymbol{X}_2 \in \mathcal{P}_F^m. \quad (1)$$

For $ij \in E$, we define the ground-truth corruption level by

$$s_{ij}^* = d(\boldsymbol{X}_{ij}, \boldsymbol{X}_{ij}^*).$$

CEMP uses cycle-consistency information to estimate these corruption levels. Recall that a 3-cycle, ijk, is a path in $G([n],E)$ containing the nodes i, j, $k \in [n]$. In our previous and current work we focus on 3-cycles for simplicity and efficient computation and refer to them just as cycles. One can extend our methods to higher-order cycles. In PS, a cycle $ijk \subseteq E$ is consistent if and only if $\boldsymbol{X}_{ij}\boldsymbol{X}_{jk} = \boldsymbol{X}_{ik}$ (equivalently, $\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\boldsymbol{X}_{ki} = \boldsymbol{I}$, where $\boldsymbol{I}$ denotes the identity matrix). We define the cycle inconsistency of the cycle ijk as

$$d_{ijk}^{\text{CEMP}} := d(\boldsymbol{X}_{ij}, \boldsymbol{X}_{ik}\boldsymbol{X}_{kj}) = 1 - \langle \boldsymbol{X}_{ij}, \boldsymbol{X}_{ik}\boldsymbol{X}_{kj}\rangle/m.$$

CEMP iteratively approximates the corruption levels as follows

$$s_{ij}^* \approx \sum_k w_{ijk} d_{ijk}^{\text{CEMP}}, \quad (2)$$

where the weights are updated at each iteration using improved estimates of the corruption levels (we omit their formulas).

The bi-invariance property of the distance in (1) implies [11]

$$d_{ijk}^{\text{CEMP}} = s_{ij}^* \text{ whenever } ik, jk \in E_g. \quad (3)$$

CEMP chooses specific weights in (2) that aim to highlight the case in (3). That is, for $ij \in E$, they are ideally close to 1 when $ik \in E_g$ and $jk \in E_g$, and close to 0 otherwise. Under some conditions, CEMP exactly recovers the corruption levels [11].

3.2. A Cycle Inconsistency Measure for PPS

For PPS, we say that a cycle ijk is consistent whenever $\boldsymbol{X}_{ij}\boldsymbol{X}_{jk} \leq \boldsymbol{X}_{ik}$, $\boldsymbol{X}_{jk}\boldsymbol{X}_{ki} \leq \boldsymbol{X}_{ji}$ and $\boldsymbol{X}_{ki}\boldsymbol{X}_{ij} \leq \boldsymbol{X}_{kj}$ (the notation $\boldsymbol{A} \leq \boldsymbol{B}$ corresponds to the component-wise inequality, i.e., $A_{ij} \leq B_{ij}$ for all indices i, j). We further motivate this definition in the supplemental material.

In order to define an inconsistency measure for PPS, we make several observations. We first formally generalize (1) and define the following conditional dissimilarity function, or divergence, where $\boldsymbol{X}_1$, $\boldsymbol{X}_2 \in \mathcal{P}^{l_1,l_2}$:

$$d(\boldsymbol{X}_1|\boldsymbol{X}_2) := 1 - \langle \boldsymbol{X}_1, \boldsymbol{X}_2\rangle/\mathsf{nnz}(\boldsymbol{X}_2),$$

where $\mathsf{nnz}(\boldsymbol{X}_2)$ is the number of nonzero elements in the matrix $\boldsymbol{X}_2$. We note that $d(\boldsymbol{X}_1|\boldsymbol{X}_2) = 0$ if and only if $\boldsymbol{X}_2 \leq \boldsymbol{X}_1$ and therefore $\boldsymbol{X}_{ik}\boldsymbol{X}_{kj} \leq \boldsymbol{X}_{ij}$ if and only if

$$\text{Tr}(\boldsymbol{X}_{ji}\boldsymbol{X}_{ik}\boldsymbol{X}_{kj}) \equiv \langle \boldsymbol{X}_{ij}, \boldsymbol{X}_{ik}\boldsymbol{X}_{kj}\rangle = \mathsf{nnz}(\boldsymbol{X}_{ik}\boldsymbol{X}_{kj}).$$

Thus, for a cycle ijk, we define $n_i = \mathsf{nnz}(\boldsymbol{X}_{ki}\boldsymbol{X}_{ij})$, $n_j = \mathsf{nnz}(\boldsymbol{X}_{kj}\boldsymbol{X}_{ji})$, $n_k = \mathsf{nnz}(\boldsymbol{X}_{ik}\boldsymbol{X}_{kj})$, $n_\Delta = \text{Tr}(\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\boldsymbol{X}_{ki})$, and conclude that ijk is cycle-consistent if and only if $3n_\Delta = n_i + n_j + n_k$. In view of this observation, we suggest the following cycle inconsistency measure for PPS:

$$d_{ijk} = 1 - 3n_\Delta/(n_i + n_j + n_k). \quad (4)$$

We further interpret this measure in the supplemental material.

Authorized licensed use limited to: University Of Minnesota Duluth. Downloaded on August 30,2023 at 03:01:47 UTC from IEEE Xplore. Restrictions apply.

3.3. The CEMP-Partial Algorithm

The CEMP algorithm for PS is motivated by (2) and (3). By replacing d_{ijk}^{CEMP} in the the CEMP algorithm by our new d_{ijk} in (4), one can obtain the CEMP-Partial algorithm for PPS, which is sketched in Algorithm 1. This algorithm iteratively updates the corruption levels according to (7) below, where the weights $w_{ijk}^{(t)}$ will be clarified below and $N_{ij} := \{k \in [n] : ik, jk \in E\}$, $ij \in E$, is the set of nodes that form cycles with the edge ij. Note that (7) is analogous to (2) in the case of PS. While in PS the aim of such a formula is to eventually yield a good estimate for the corruption levels, in PPS we only aim to cluster uncorrupted and corrupted edges with low (close to zero) and sufficiently high corruption levels, respectively; we show that this is indeed possible in §4. Before the iterations, the corruption levels are initialized in (5) as a similar average but with uniform weights. Given the corruption levels of the previous iteration, the weights of the current iteration are computed by (6), where β_t is a fixed parameter at iteration t. This formula aims to assure that $w_{ijk}^{(t)}$ is large (close to 1) whenever both ik and jk are good and close to zero otherwise, so that the estimated $s_{ij}^{(t+1)}$ in (6) is approximately an average of only those d_{ijk}'s with good edges ik and jk. Such a behavior of $w_{ijk}^{(t)}$ occurs when the estimates of the corruption levels s_{ij} is close to zero when ij is a good edge and sufficiently far from zero otherwise. In fact, alternatively updating both the weights and the corruption levels aims to result in such a property (see §4).

Algorithm 1 CEMP-Partial

```
Input: $\{d_{ijk}\}_{ij \in E, k \in N_{ij}}$ (see (4)), number of iterations $T$,
    increasing $\{\beta_t\}_{t=1}^{T}$
  Steps:
      $s_{ij}^{(0)} = \frac{1}{|N_{ij}|} \sum_{k \in N_{ij}} d_{ijk}$  for  $ij \in E$        (5)
  for $t = 0 : T-1$ do
      $w_{ijk}^{(t)} = \exp\left(-\beta_t \left(s_{ik}^{(t)} + s_{jk}^{(t)}\right)\right)$, $k \in N_{ij}$, $ij \in E$        (6)
      $s_{ij}^{(t+1)} = \frac{1}{Z_{ij}^{(t)}} \sum_{k \in N_{ij}} w_{ijk}^{(t)} d_{ijk}$ for  $ij \in E$        (7)

      where $Z_{ij}^{(t)} = \sum_{k \in N_{ij}} w_{ijk}^{(t)}$ is a normalization factor
  end for
Output: $\hat{s}_{ij} := s_{ij}^{(T)}$  for  $ij \in E$
```

3.4. MatchFAME

We propose MatchFAME that aims to address the main challenges of PPS (see §1). MatchFAME combines CEMP-Partial with a weighted PPM method. The original (unweighted) PPM is an iterative procedure that aims to minimize the least squares energy $\sum_{ij \in E} \|\boldsymbol{P}_i \boldsymbol{P}_j^\top - \boldsymbol{X}_{ij}\|_F^2$, under the constraint $\boldsymbol{P}_i \in \mathcal{P}_F^m$ for $i \in [n]$. Given the estimated absolute permutations for the different nodes at the t-th iteration, $\{\boldsymbol{P}_i^{(t)}\}_{i \in [n]}$, PPM estimates each permutation on node i in the next iteration as

$$\boldsymbol{P}_i^{(t+1)} = \textbf{Proj}\left(\sum_{j \in N_i} \boldsymbol{X}_{ij} \boldsymbol{P}_j^{(t)} / |N_i|\right), \tag{8}$$

where N_i denotes the neighboring nodes of i, and **Proj** is the projection onto the space of permutations, which can be computed by the Hungarian algorithm [14]. Intuitively, at iteration t of PPM, each $j \in N_i$ proposes the "local" estimate of $\boldsymbol{P}_i^*$: $\boldsymbol{X}_{ij} \boldsymbol{P}_j^{(t)}$, and $\boldsymbol{P}_i$ in the new iteration is updated by the average of these local estimates followed by a projection. However, these local estimates are only accurate when $\boldsymbol{X}_{ij} \approx \boldsymbol{X}_{ij}^*$ and $\boldsymbol{P}_j^{(t)} \approx \boldsymbol{P}_j^*$ for all $j \in N_i$. This makes PPM sensitive to both initialization and edge corruption, and thus a naive generalization of PPM to partial permutations is not sufficient to handle the PPS challenges described in §1.

To address the first PPS challenge of nonuniform corruption (see §1), we assign a weight w_{ij} to each $ij \in E$, where w_{ij} depends on $\hat{s}_{ij}$, the estimated corruption level by CEMP-Partial, and a parameter γ, which measures the confidence of the estimated corruption levels, as follows: $w_{ij} = \exp(-\gamma \hat{s_{ij}})$. We can thus implement a weighted PPM iteration for each $i \in [n]$:

$$\boldsymbol{P}_i^{(t+1)} = \textbf{Proj}\left(\sum_{j \in N_i} \tilde{w}_{ij} \boldsymbol{X}_{ij} \boldsymbol{P}_j^{(t)}\right), \tag{9}$$

where $\boldsymbol{P}_i^{(t)}, \boldsymbol{P}_i^{(t+1)} \in \mathcal{P}^{m_i, m}$, $\tilde{w}_{ij} = w_{ij} / \sum_{j \in N_i} w_{ij}$ are the normalized weights and **Proj** is the heuristic and fast projection onto partial permutations described in [13]. In such a way, the projected power iterations will focus on the clean edges and thus largely mitigate the sensitivity of standard PPM towards nonuniform topology of the corrupted subgraph, and nonuniform distribution of the corrupted partial permutations [20]. We remark that given the ideal weights $w_{ij} = \mathbf{1}_{\{ij \in E_g\}}$, where $\mathbf{1}$ is the indicator function, the ground truth permutations, $\{\boldsymbol{P}_i^*\}_{i \in [n]}$, form a fixed point of our weighted PPM.

The second PPS challenge of highly-demanding computation (see §1) arises in PPM if it is initialized by Spectral or another standard PPS algorithm. In order to resolve this issue, we follow [19] and initialize our solution using a minimum spanning tree (MST), which depends on the output of CEMP-Partial. Specifically, we build a weighted graph where edge weights are the estimated corruption levels by CEMP-Partial. An MST is then extracted from the weighted graph. Note that it has the lowest average corruption levels among all other spanning trees. We then use it to initialize the absolute permutations. We first arbitrarily assign to the root node the partial permutation $\boldsymbol{I}_{m_i \times m}$, which is an $m_i \times m$ matrix whose diagonal elements are 1 and the rest are 0. We subsequently multiply relative permutations along the MST, namely applying

Authorized licensed use limited to: University Of Minnesota Duluth. Downloaded on August 30,2023 at 03:01:47 UTC from IEEE Xplore. Restrictions apply.

$\boldsymbol{P}_j^{(t)} = \textbf{Proj}(\boldsymbol{X}_{ji}\boldsymbol{P}_i^{(t)})$ from the root to the leaves. We note that computation of the MST only uses the small $n\times n$ adjacency matrix of the graph, unlike spectral initialization that involves the eigenvalue decomposition of a huge $M \times M$ matrix. As a result, our initialization is much faster and memory efficient, which breaks the computational bottleneck of standard PPM.

The last challenge of PPS is the uneven dimension and sparsity level of partial permutations. The main hurdle was the generalization of CEMP to PPS (see §3.3). This way, the weights for our weighted PPM can be reliably estimated. Another issue that arises due to this challenge is that the MST initialization may be too sparse at the end of the spanning tree if some partial relative permutations in $\{\boldsymbol{X}_{ij}\}_{ij\in E}$ are extremely sparse. In some extreme cases, some columns of the estimated absolute permutations are zero. Therefore, after initializing the absolute permutations, we check if there is a zero column for the block column matrix of the estimated absolute permutations. If it is the case, we randomly fill one of the elements with 1. In some special cases, where denser matrices are needed, one may fill instead zero rows (see supplemental material).

The full description of MatchFAME is in Algorithm 2. As is common in SfM, MatchFAME outputs refined and consistent keypoint matches: $\boldsymbol{Z}_{ij} := \hat{\boldsymbol{P}}_i\hat{\boldsymbol{P}}_j^\top$ for any $ij\in E$.

Algorithm 2 MatchFAME

```
Input: CEMP-Partial output, $\{\hat{s}_{ij}\}_{ij\in E}$, pairwise matching matrix $\boldsymbol{X}$, parameter $\gamma$, number of iterations $t_0$
  Steps:
  Form weights on $E$: $w_{ij} = \hat{s}_{ij}$ for $ij\in E$
  Find the MST of the weighted graph $G([n],E,\{w_{ij}\}_{ij\in E})$
  For the root of tree at node $i_0\in[n]$: $\boldsymbol{P}_{i_0}^{(0)} = \boldsymbol{U}_{m_i\times m}$
  Propagate along MST: $\boldsymbol{P}_j^{(0)} = \textbf{Proj}(\boldsymbol{X}_{ji}\boldsymbol{P}_i^{(0)})$; if a column of $[\boldsymbol{P}_j^{(0)}]_{j\in[n]}$ is $\mathbf{0}$, randomly assign 1 to one of its elements
  Reset graph weights: $w_{ij} = e^{-\gamma s_{ij}}$ for $ij\in E$
  Set $t=0$
  while $t<t_0$ and $\boldsymbol{P}_i^{(t)} \neq \boldsymbol{P}_i^{(t-1)}$ do
    Compute $\boldsymbol{P}_i^{(t+1)}$ by (9) for $i\in[n]$
    $t=t+1$
  end while
Output: $\boldsymbol{Z}_{ij} = \boldsymbol{P}_i^{(t_0+1)}\boldsymbol{P}_j^{(t_0+1)\top}$ for $ij\in E$
```

The default parameters for CEMP-Partial are $T = 25$ and $\beta_t = \min\{1.2^t, 40\}$. The default parameters for MatchFAME are $t_0 = 60$ and $\gamma = 4$ (for the noiseless synthetic data we use $\gamma = 20$ to approach exact recovery).

3.5. Time and Space Complexity

Recall that n is the number of images and M is the total number of 2D keypoints (which is different from m), so that the average number of 2D keypoints in each image is M/n. Let an_E denote the number of edges in the viewing graph. The time and space complexities of CEMP-Partial are $O(Mn_E)$ and $O(nn_E)$ respectively. Let $\boldsymbol{P}$ be the $M\times m$ block matrix whose i-th block is $\boldsymbol{P}_i$, $\boldsymbol{X}$ be the $M\times M$ block matrix whose (i,j)-th block is $\boldsymbol{X}_{ij}$. The power iterations in (9), considering all $i \in [n]$, can be equivalently viewed as a multiplication between a weighted sparse matrix $\boldsymbol{X}$ and a sparse matrix $\boldsymbol{P}$. Using the facts that there are at most n nonzero elements in each column of $\boldsymbol{P}$ and each row of $\boldsymbol{X}$ and each row of $\boldsymbol{P}$ has at most 1 nonzero element, the time and space complexities for the power iterations are $O(nM)$ and $O(nm)$, respectively. Note that PPM requires an additional projection onto the set of partial permutations, whose time complexity is $O(Mm)$ and space complexity $O(nm)$. The time complexity for finding MST is $O(n_E \log n)$ and its space complexity is $O(n)$. The time complexity of multiplying matrices along the spanning tree is $O(n^2)$ and it requires no additional memory. To sum up, since $n < n_E$, MatchFAME requires $O(M\cdot\max(m,n_E))$ time and $O(n\cdot\max(m,n_E))$ memory. In comparison, Spectral computes the top m eigenvectors of $\boldsymbol{X}$, which is commonly solved by the power method. This method requires the iterative multiplication of $\boldsymbol{X}$ with the updated $M \times m$ dense eigenmatrix, whose time and space complexities are $O(Mmn)$ and $O(Mm)$, respectively, which is much larger than that of MatchFAME.

4. Theoretical Guarantees for CEMP-Partial

We assume the adversarial corruption model for PPS (see §2). We show that the estimated corruption levels by CEMP-Partial at good edges converge to 0 linearly and uniformly. Moreover, we show that the estimated corruption levels at the bad edges, can be separated from the ones at the good edges.

Definitions: For $ij \in E$, define $G_{ij} := \{k\in[n] : ik,jk\in E_g\}$ and associate $k \in G_{ij}$ with the cycle ijk. We refer to the elements of G_{ij} as good cycles (with respect to ij). Let

$$\lambda := 1 - \min_{ij} |G_{ij}|/|N_{ij}|.$$

Some of the following definitions, which lead to the notion of s_{ij}^*, are demonstrated in Figure 1. For any image i, let $I_i = \{p_{i,j}\}_{j=1}^{m_i}$ denote the set of its 2D keypoints. Let $h : \cup_{i=1}^n I_i \to [m]$ map 2D keypoints to universal keypoints such that for any $i\in[n]$ and $p_{i,i'} \in I_i$, $h(p_{i,i'})$ is the index of the 3D keypoint, i.e., $P_i^*(i', h(p_{i,i'})) = 1$. For $ij \in E$, let $U_{ij} = h(I_i)\cup h(I_j)$. We partition U_{ij} into U_{ij}^{good} and U_{ij}^{bad}. The set U_{ij}^{good} contains the keypoints in U_{ij} that match the ground-truth keypoints or match no keypoint when no ground-truth matching exists. That is, $k\in h(I_i)\cap h(I_j)$ is in U_{ij}^{good} whenever there exists $p_{i,a}\in I_i$ and $p_{j,b} \in I_j$ such that $\boldsymbol{P}_i^*(a,k) = \boldsymbol{P}_j^*(b,k) = \boldsymbol{X}_{ij}(a,b) = 1$. Furthermore, $k \in h(I_i)\setminus h(I_j)$ (or $k \in h(I_j)\setminus h(I_i)$) is in U_{ij}^{good} whenever there are no $p_{i,a}\in I_i$ and $p_{j,b}\in I_j$ such that $\boldsymbol{P}_i^*(a,k) = \boldsymbol{X}_{ij}(a,b) = 1$ (or $\boldsymbol{P}_j^*(b,k) = \boldsymbol{X}_{ij}(a,b) = 1$). Similarly, $U_{ij}^{\text{bad}} := U_{ij}\setminus U_{ij}^{\text{good}}$ is the set of keypoints in U_{ij} that match wrong keypoints in the other image, or match no keypoint if a

Authorized licensed use limited to: University Of Minnesota Duluth. Downloaded on August 30,2023 at 03:01:47 UTC from IEEE Xplore. Restrictions apply.

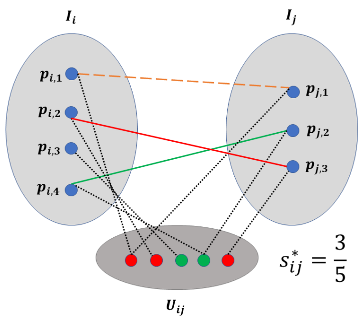

Figure 1. An illustration of s^*_{ij} and related definitions. The sets I_i and I_j are of 2D keypoints in images i and j, respectively, and the set U_{ij} is of the corresponding universal keypoints in images i and j. Green lines denote the good keypoint matches, red lines denote the bad keypoint matches, dashed orange lines denote the missing keypoint matches, and black dotted lines denote the correspondence between 2D and 3D keypoints (i.e., the h function). Green dots in U_{ij} represent elements of U^{good}_{ij} and red dots in U_{ij} represent elements of U^{bad}_{ij}. We note that $m_{ij} = 5$, $m^{\text{good}}_{ij} = 2$, $m^{\text{bad}}_{ij} = 3$ and $s^*_{ij} = \frac{3}{5}$.

ground-truth match exists. For any $ij \in E$, define $m_{ij} = |U_{ij}|$, $m^{\text{bad}}_{ij} = |U^{\text{bad}}_{ij}|$ and note that $s^*_{ij} = m^{\text{bad}}_{ij}/m_{ij}$.

In order to guarantee that the separation problem is well-posed, we need to ensure two different types of conditions. The first is that there are sufficiently many good cycles. We ensure this condition by bounding λ (similarly to [11]). The second is that sufficiently many good matches exist. Indeed, since partial permutations can be very sparse and even zero matrices such a condition is necessary. For this purpose, we formulate the following cycle-verfiability condition (we further interpret it and clarify its name in the supplemental material). It uses a parameter p_v that expresses the proportion of verifiability.

Definition 1. Given $p_v \in (0, 1]$, a graph (V, E) is p_v-cycle verifiable if for any $ij \in E$ there are at least $p_v|G_{ij}|$ good cycles w.r.t. ij such that for each such cycle, ijk, the following property holds: if $a \in I_i \cup I_j$, then there exists $b \in I_k$ that matches a (i.e., if $a = p_{i,r}$ and $b = p_{k,t}$, then $\boldsymbol{X}_{ik}(r,t) = 1$).

Formulation of the Main Theorem:

Theorem 1. If G is p_v-cycle verifiable and $s^{(t)}_{ij}$ is computed by CEMP-Partial with $\beta_0 \le \frac{1}{2\lambda}$ and $\beta_{t+1} = r\beta_t$, where $\lambda < 1 + \frac{3em}{p_v} - \sqrt{\frac{3em}{p_v}(2 + \frac{3em}{p_v})}$ and $1 < r < \frac{(1-\lambda)^2 p_v}{6em\lambda}$, then $\forall t > 0$

$$\forall ij \in E_g \;\; s^{(t)}_{ij} \le \frac{1}{2\beta_0 r^t} \;\text{ and }\; \forall ij \in E_b \;\; s^{(t)}_{ij} \ge \frac{p_v}{3e}(1-\lambda)s^*_{ij}.$$

Theorem 1 guarantees exact separation between clean and corrupted edges in a worst-case scenario setting. Unlike previous PPS works [6, 16], our theory is completely deterministic and does not rely on the assumptions of the underlying distribution of partial permutations. Its deterministic conditions guarantee well-posedness of the separation problem.

5. Numerical Experiments

5.1. Synthetic Data Experiments

We test MatchFAME, MatchEIG [13], Spectral [16], and PPM [5] on synthetic datasets generated by two different models described in §5.1.1 and §5.1.2. The underlying graph in both cases is generated by an Erdös-Rényi model, $G(n,p)$, with probability of edge connection $p = 0.5$. The inclusion of a keypoint in an image I is an independent event from the rest of the keypoints that occurs with probability $p_I = 0.8$. In both cases, the number of images is $n = 100$ and the universe size is $m = 20$ (except for the runtime experiments in §5.1.2). Since m is typically unknown in practice, all algorithms use the following estimate for it: $\hat{m} = 2\lceil M/n \rceil$. In order to demonstrate near exact separation, we use $\gamma = 20$ for the synthetic data, where the rest of the default parameters are as specified in §3.4.

After generating the graph $G([n], E)$ by the $G(n, p)$ model, we also randomly generate an $nm \times m$ ground truth image-universe matching matrix $\boldsymbol{P}^{\text{full}}$. Each of its n blocks is obtained by randomly permuting columns of $\boldsymbol{I}_{m\times m}$. We further compute $\boldsymbol{X}^{\text{full}} = \boldsymbol{P}^{\text{full}}\boldsymbol{P}^{\text{full}\top}$. The models below further corrupt $\boldsymbol{X}^{\text{full}}$ and result in modified matrix $\hat{\boldsymbol{X}}^{\text{full}}$. We then generate the keypoint indices, $\{I_i\}_{i=1}^n$, as follows: For each $i \in [n]$, we independently generate m i.i.d. $\sim B(1,0.8)$ random variables and let I_i be the set of indices of random variables with output 1. For each $i \in [n]$ and the ith $m \times m$ block of $\boldsymbol{P}^{\text{full}}$, we keep the rows with indices in I_i and discard the rest. This results in a block $\boldsymbol{P}^*_i$ of size $m_i \times m$. The resulting block matrix of all modified blocks is of size $M \times m$ and denoted by $\boldsymbol{P}^*$. We further set $\boldsymbol{X}^* = \boldsymbol{P}^*\boldsymbol{P}^{*T}$. For the ij-th $m \times m$ block of $\hat{\boldsymbol{X}}^{\text{full}}$, we keep its rows that appear in I_i and columns that appear in I_j and discard the rest. This results in a block of size $m_i \times m_j$. We stack these blocks to form the $M \times M$ matrix $\boldsymbol{X}$.

Let $\odot$ denote the element-wise product and $\boldsymbol{Z}_{ij}$ denote the ij-th block of the output $\boldsymbol{Z}$. Precision and recall were respectively computed as follows:

$$\sum_{ij\in E_b} \langle \boldsymbol{X}^*_{ij} \odot \boldsymbol{X}_{ij}, \boldsymbol{Z}_{ij} \odot \boldsymbol{X}_{ij} \rangle / \sum_{ij\in E_b} \|\boldsymbol{Z}_{ij} \odot \boldsymbol{X}_{ij}\|_F^2 \;\text{ and}$$

$$\sum_{ij\in E_b} \langle \boldsymbol{X}^*_{ij} \odot \boldsymbol{X}_{ij}, \boldsymbol{Z}_{ij} \odot \boldsymbol{X}_{ij} \rangle / \sum_{ij\in E_b} \|\boldsymbol{X}^*_{ij} \odot \boldsymbol{X}_{ij}\|_F^2.$$

Since in SfM precision is more important than recall [13], we find an algorithm superior to another if it achieves significantly higher precision with almost equal or higher recall.

5.1.1 Data Generated by the LBC and LAC Models

We extend the Local Biased Corruption (LBC) and Local Adversarial Corruption (LAC) models of [20] to PPS. Both models introduce nonuniform corruption concentrated in some clusters, where for some nodes, most of their neighboring edges are corrupted. LBC assumes that bad edges are also

Authorized licensed use limited to: University Of Minnesota Duluth. Downloaded on August 30,2023 at 03:01:47 UTC from IEEE Xplore. Restrictions apply.

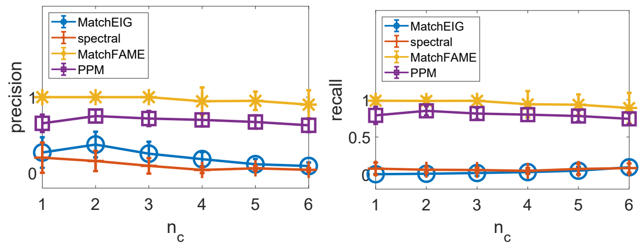

Figure 2. Average precision and recall for the LBC model

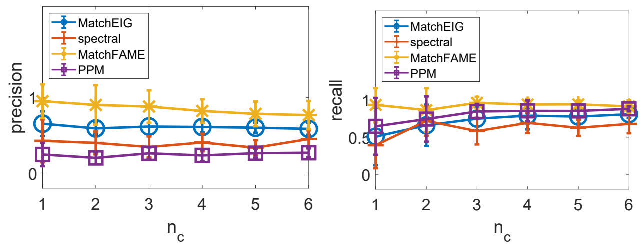

Figure 3. Average precision and recall for the LAC model

cycle-consistent, so they behave like good edges. In addition, it uses a sample-rejection procedure so that the distribution of the corrupted partial permutations deviates from the uniform distribution. LAC is even more malicious, it corrupts edges in a way that would seem the absolute permutations of the selected nodes are perturbed versions of $\boldsymbol{I}_{m_i,m}$.

We let $\{\boldsymbol{P}_i^c\}_{i\in[n]}$ be i.i.d. sampled from the Haar measure on $\mathcal{P}_{\text{full}}^{m\times m}$. Starting from $\hat{\boldsymbol{X}}^{\text{full}} = \boldsymbol{X}^{\text{full}}$, for both models we independently sample n_c nodes as our corruption seed nodes. For each corruption seed node, we independently corrupt its associated edges with probability 0.9 for LBC and 0.6 for LAC. Each $ij \in E_b$ in LBC is corrupted as follows:

$$\hat{\boldsymbol{X}}_{ij}^{\text{full}} = \begin{cases} \hat{\boldsymbol{X}}_{ij}^{\text{full}} \sim \text{Haar}(\mathcal{P}_{\text{full}}^{m\times m}), & \text{if } \langle \boldsymbol{P}_i^c \boldsymbol{P}_j^{c\top}, \boldsymbol{X}_{ij}^{\text{full}} \rangle > 1; \\ \boldsymbol{P}_i^c \boldsymbol{P}_j^{c\top}, & \text{otherwise.} \end{cases}$$

In LAC, for $ij \in E_b$, $\hat{\boldsymbol{X}}_{ij}^{\text{full}} = \boldsymbol{Q}_{ij}^c \boldsymbol{P}_j^{\text{full}\top}$, where $\boldsymbol{Q}_{ij}^c$ is obtained by randomly permuting 3 of the columns of $\boldsymbol{I}_{m\times m}$.

The resulting precision and recall are reported in Figure 2 for the LBC model and Figure 3 for the LAC model. We note that for both models, MatchFAME recovers almost exactly all bad edges, where other algorithms have lower errors and are not sufficiently close to exact recovery. MatchFAME also obtains the highest recall scores, which are close to 1.

5.1.2 Data Generated by the Uniform Corruption Model

We test MatchFAME and competing models on data generated from the uniform corruption model (UCM). In this model, an edge is randomly selected with probability q and then corrupted as follows: $\hat{\boldsymbol{X}}_{ij}^{\text{full}} \sim \text{Haar}(\mathcal{P}_{\text{full}}^{m\times m})$. For unselected edges: $\hat{\boldsymbol{X}}_{ij}^{\text{full}} = \boldsymbol{X}_{ij}^{\text{full}}$. We let q range between 0.5 and 0.9. Figure 4

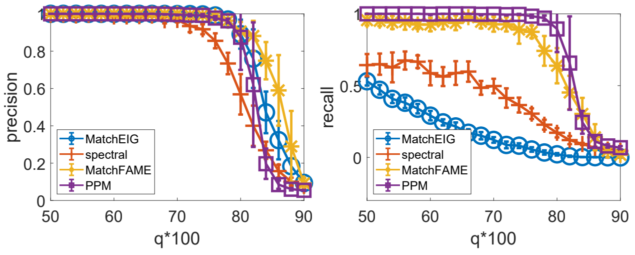

Figure 4. Average precision and recall for UCM

Algorithm	$m=500$	$m=1000$	$m=2000$
MatchFAME	1	2	27
MatchEIG	19	194	1297

Table 1. Runtime (seconds) for UCM with $n=20$.

Algorithm	$n=300$	$n=500$	$n=700$
MatchFAME	86	304	826
MatchEIG	116	1099	>5000

Table 2. Runtime (seconds) for UCM with $m=20$.

reports the precision and recall of the different methods. We observe that for UCM, the precision of MatchFAME decreases the slowest among all algorithms. We observe that MatchFAME and PPM have similar high recall, while MatchEIG and Spectral have relatively lower recall.

For runtime comparison, we compared MatchFAME with MatchEIG, which is the fastest PPS method. Table 1 fixes $n=20$ and reports results for different values of m and Table 2 fixes $m = 20$ and reports results for different values of n (experiments stopped when the time was larger than 5000 seconds). Clearly, MatchFAME is significantly faster than MatchEIG for large m and sufficiently large n.

5.2. Real Data Experiments

We test MatchFAME on the Photo Tourism dataset [26]. This large-scale dataset contains 14 sets of images for stereo reconstruction. The number of images in each dataset ranges from 230 to 2226. Given initial keypoint matches obtained by [17], we form our pairwise matching matrix and estimate the universe size with $\hat{m} = 16\lceil M/n \rceil$. We apply MatchFAME with its default parameters. After getting the output keypoint matching, we use RANSAC to estimate the fundamental matrices for each edge. If an edge has less than 16 remaining keypoint matches, we remove this edge. Then we decompose the resulting fundamental matrices and feed the estimated rotations and translations to the LUD [15] camera pose solver to obtain the final estimate of absolute rotations and translations. Note that LUD extracts the largest parallel rigid component of the remaining graph, therefore removing edges can cause loss of cameras. We compare the average and median translation error, average and median rotation error, and runtime to the original pipeline of LUD. We

Authorized licensed use limited to: University Of Minnesota Duluth. Downloaded on August 30,2023 at 03:01:47 UTC from IEEE Xplore. Restrictions apply.

Algorithms			LUD						MatchFAME+LUD						
Dataset	n	M	n_{LUD}	$\hat{e}_R$	$\tilde{e}_R$	$\hat{e}_T$	$\tilde{e}_T$	$T_{\text{total,LUD}}$	n_{FAME}	$\hat{e}_R$	$\tilde{e}_R$	$\hat{e}_T$	$\tilde{e}_T$	T_{FAME}	T_{total}
Alamo	570	606963	557	20.81	16.88	8.00	5.23	7945.9	494	**17.50**	**14.19**	**6.99**	**4.58**	6272.0	16523.0
Ellis Island	230	178324	223	2.14	1.15	22.99	22.82	1839.2	216	**1.79**	**1.02**	**22.63**	**21.57**	341.5	2115.7
Gendarmenmarkt	671	338800	652	**40.14**	9.30	**38.55**	**18.33**	3527.7	574	40.20	**9.20**	42.29	21.65	662.2	3960.2
Madrid Metropolis	330	187790	315	13.49	9.58	14.10	6.81	1579.8	266	**10.21**	**6.18**	**9.90**	**4.42**	179.7	1663.2
Montreal N.D.	445	643938	439	2.55	1.06	1.51	0.66	5078.9	389	**1.45**	**0.79**	**1.16**	**0.60**	4021.0	8666.4
Notre Dame	547	1345766	545	3.72	**1.44**	1.45	0.41	11315.2	526	**3.19**	1.48	**1.27**	**0.40**	27222.4	39189.5
NYC Library	313	259302	306	3.99	2.14	6.89	2.72	1495.9	270	**2.96**	**1.94**	**6.40**	**2.43**	219.9	2112.5
Piazza Del Popolo	307	157971	300	7.04	4.05	6.68	2.33	1989.9	241	**1.39**	**0.83**	**2.26**	**1.33**	309.6	2519.3
Piccadilly	2226	1278612	2015	8.05	3.82	5.47	2.98	21903.3	1479	**4.96**	**2.86**	**3.96**	**2.16**	26170.1	48157.3
Roman Forum	995	890945	971	6.64	5.02	12.67	5.60	4858.0	733	**5.63**	**4.29**	**11.73**	**5.55**	1548.3	6955.3
Tower of London	440	474171	431	**6.89**	4.29	21.47	6.85	1759.2	359	7.13	**4.19**	**13.54**	**6.32**	282.5	2040.4
Union Square	733	323933	663	10.40	6.70	15.27	11.14	1950.6	451	**8.26**	**5.29**	**10.71**	**8.66**	224.9	2064.9
Vienna Cathedral	789	1361659	758	6.45	3.10	14.18	8.12	10866.0	631	**4.21**	**2.02**	**11.90**	**6.90**	21082.7	30066.5
Yorkminster	412	525592	407	**4.25**	2.71	6.45	3.68	2267.3	359	4.26	**2.56**	**6.41**	**3.37**	586.4	2936.5

Table 3. Performance on the Photo Tourism database: n and M are the number of nodes and keypoints, respectively; n_{LUD} and n_{FAME} are the remaining number of cameras after the LUD pipeline and our pipeline, respectively; $\hat{e}_R$ $\tilde{e}_R$ indicate mean and median errors of absolute camera rotations in degrees, respectively; $\hat{e}_T$ $\tilde{e}_T$ indicate mean and median errors of absolute camera translations in meters, respectively; T_{FAME}, $T_{\text{total,LUD}}$ and T_{total} are the runtime of MatchFAME, the total runtime of LUD pipeline and the total runtime of our pipeline, respectively (in seconds).

remark that we did not compare with other PPS algorithms as the ones that were available at the time of the submission were not scalable and could not handle the Photo Tourism dataset.

Table 3 reports results for both the LUD pipeline and the incorporation of MatchFAME within the LUD pipeline. We consider improvement over the LUD result when we obtain a smaller error on at least 3 of the 4 error statistics. MatchFAME is successful in improving the estimates of translation and rotation without significant loss of cameras in 13 of the 14 datasets. The most significant improvement is on Piazza Del Popolo, where our mean and median rotation error decreased by 80.3% and 79.5% respectively. For most other datasets, the improvement was not marginal. Indeed, in 12 of these 13 datasets, at least one error statistic decreased by more than 15%. The only dataset we didn't improve is Gendarmenmarkt, which has very high error because of its symmetric buildings. MatchFAME removes at most 32.0% of the total cameras. The remaining cameras are sufficient for 3D reconstruction since the Photo Tourism cameras are sampled densely. Therefore our pipeline will not cause significant loss of quality of 3D reconstruction. That is, MatchFAME is able to remove cameras with erroneous keypoints without losing the stereo reconstruction power.

On larger datasets, the total time of MatchFAME is 2 to 3 times larger than that of the original LUD pipeline. We thus find MatchFAME scalable. On smaller datasets, such as Union Square and Madrid Metropolis, MatchFAME consumes much less time in comparison to the original LUD pipeline.

6. Conclusion

We develop MatchFAME, a robust, fast, accurate and memory efficient PPS method. For this purpose we first developed CEMP-Partial for corruption estimation in PPS and theoretically guaranteed it under adversarial corruption. In doing this, we were able to overcome nontrivial challenges of the PPS problem. We also proposed an efficient weighted PPM method that utilizes the output of CEMP-Partial and, in particular, does not require spectral initialization. MatchFAME overcomes the three major challenges of PPS: nonuniform corruption, uneven dimensions and sparsity levels, and the large problem size. Synthetic and real data experiments demonstrate the superior precision of MatchFAME over existing standard methods and its scalability to large datasets due to sparse matrix operations. Our method also has some limitations. For example, our MST initialization is quite heuristic and may produce very sparse initialization. Instead, one may consider using minimum-K spanning trees and initialize the solutions by aggregating the initialization from the different spanning trees. Moreover, our weights for PPM depend on a parameter γ and we plan to explore in future work the optimal assignment of this parameter, using the estimates of the corruption levels. Furthermore, our theory is currently limited to CEMP-Partial and to adversarial corruption with certain assumptions. Nevertheless, it demonstrates how to handle some challenges that are unique to PPS. We plan to further extend it. We will first explore other corruption models, such as UCM, where we expect stronger convergence guarantees. We also plan to further develop theory for PPM, in particular, for UCM, and hopefully establish a more complete theory for MatchFAME.

Acknowledgement

This work was supported by NSF awards 1821266, 2124913.

Authorized licensed use limited to: University Of Minnesota Duluth. Downloaded on August 30,2023 at 03:01:47 UTC from IEEE Xplore. Restrictions apply.

References

[1] Federica Arrigoni, Andrea Fusiello, Elisa Ricci, and Tomas Pajdla. Viewing graph solvability via cycle consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5540–5549, 2021.

[2] Federica Arrigoni, Eleonora Maset, and Andrea Fusiello. Synchronization in the symmetric inverse semigroup. In *International conference on image analysis and processing*, pages 70–81. Springer, 2017.

[3] Tolga Birdal and Umut Simsekli. Probabilistic permutation synchronization using the riemannian structure of the birkhoff polytope. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[4] Avishek Chatterjee and Venu Madhav Govindu. Efficient and robust large-scale rotation averaging. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 521–528, 2013.

[5] Yuxin Chen and Emmanuel J. Candès. The projected power method: an efficient algorithm for joint alignment from pairwise differences. *Comm. Pure Appl. Math.*, 71(8):1648–1714, 2018.

[6] Yuxin Chen, Leonidas J. Guibas, and Qi-Xing Huang. 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*, pages 100–108, 2014.

[7] Olof Enqvist, Fredrik Kahl, and Carl Olsson. Non-sequential structure from motion. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 264–271. IEEE, 2011.

[8] Richard I. Hartley, Khurrum Aftab, and Jochen Trumpf. 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*, pages 3041–3048, 2011.

[9] Qi-Xing Huang and Leonidas J. Guibas. Consistent shape maps via semidefinite programming. *Comput. Graph. Forum*, 32(5):177–186, 2013.

[10] Vahan Huroyan. *Mathematical Formulations, Algorithm and Theory for Big Data Problems*. PhD thesis, University of Minnesota, 2018.

[11] Gilad Lerman and Yunpeng Shi. Robust group synchronization via cycle-edge message passing. *arXiv preprint arXiv:1912.11347*, 2019.

[12] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[13] Eleonora Maset, Federica Arrigoni, and Andrea Fusiello. Practical and efficient multi-view matching. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4578–4586, 2017.

[14] James Munkres. Algorithms for the assignment and transportation problems. *J. Soc. Indust. Appl. Math.*, 5:32–38, 1957.

[15] Onur Özyesil and Amit Singer. Robust camera location estimation by convex programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2674–2683, 2015.

[16] Deepti Pachauri, Risi Kondor, and Vikas Singh. Solving the multi-way matching problem by permutation synchronization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1860–1868. Curran Associates, Inc., 2013.

[17] Soumyadip Sengupta, Tal Amir, Meirav Galun, Tom Goldstein, David W. Jacobs, Amit Singer, and Ronen Basri. A new rank constraint on multi-view fundamental matrices, and its application to camera location recovery. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, Hawaii, USA, June 22-25, 2017*, pages 4798–4806, 2017.

[18] Tianwei Shen, Siyu Zhu, Tian Fang, Runze Zhang, and Long Quan. Graph-based consistent matching for structure-from-motion. In *European Conference on Computer Vision*, pages 139–155. Springer, 2016.

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

[20] Yunpeng Shi, Shaohan Li, and Gilad Lerman. Robust multi-object matching via iterative reweighting of the graph connection laplacian. *Advances in Neural Information Processing Systems*, 2020-December, 2020.

[21] Yunpeng Shi, Shaohan Li, Tyler Maunu, and Gilad Lerman. Scalable cluster-consistency statistics for robust multi-object matching. In *2021 International Conference on 3D Vision (3DV)*, pages 352–360. IEEE, 2021.

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

[23] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM Siggraph 2006 Papers*, pages 835–846. 2006.

[24] Lanhui Wang and Amit Singer. Exact and stable recovery of rotations for robust synchronization. *Information and Inference*, 2013.

[25] Qianqian Wang, Xiaowei Zhou, and Kostas Daniilidis. Multi-image semantic matching by mining consistent features. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018.

[26] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*, pages 61–75, 2014.

[27] Christopher Zach, Manfred Klopschitz, and Marc Pollefeys. 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*, pages 1426–1433, 2010.

[28] Xiaowei Zhou, Menglong Zhu, and Kostas Daniilidis. Multi-image matching via fast alternating minimization. In *IEEE International Conference on Computer Vision, ICCV 2015*, 2015.

Authorized licensed use limited to: University Of Minnesota Duluth. Downloaded on August 30,2023 at 03:01:47 UTC from IEEE Xplore. Restrictions apply.

Supplemental Material

A. Additional Experiments on the EPFL dataset

We test MatchFAME on the 6 EPFL datasets following the experimental setup of [13]. Each dataset includes 8 to 30 images, unlike the large number of images in the Photo Tourism datasets. Given each dataset, we generate and refine the initial keypoint matches with the same procedure introduced in [13]. We follow their convention and estimate the universe size with $\hat{m}=2\lceil M/n\rceil$. We implement MatchFAME with its default parameters, though with two changes described below. Indeed, the EPFL dataset contains a lot of noisy edges and thus the weights produced by PPM within the original MatchFAME algorithm are often small. Furthermore, note that $\mathbf{Proj}$ in (9) is not scale invariant and that the resulting small weights may lead to overly sparse refined matches. Therefore, we slightly changed the implementation of MatchFAME to overcome this issue. First, in order to obtain a dense initialization of partial permutations using MST, instead of assigning 1 to a random element for each zero column, we assign 1 to a random element for each zero row. Since the number of rows is larger than the number of columns, this modification results in a denser initialization of $[\boldsymbol{P}_j^{(0)}]_{j\in[n]}$ than that of the original MatchFAME. Second, to make sure that the final output is also sufficiently dense, we drop the step of the weights' normalization within the PPM iterations, which is described below (9) (this will increase the overall scale of the edge weights and thus the projected matrix is expected to be denser). We remark that these two changes help alleviate the over-sparseness of the final output and ends up with a higher ratio between the number of refined matches and the number of initial matches, which we denote by $\#M$.

In addition to this version of MatchFAME, we also test Spectral, MatchEIG and MatchALS with the same setting as [13]. Note that the 'ground truth' is obtained by estimating the projection distance of key points on the epipolar line instead of labeling by hand. Therefore the recall score is not a good benchmark on real data. We thus only report the resulting precision, number of remaining edges and runtime in Table 4.

MatchFAME achieves the highest precision of all methods in all datasets. Observing $\#M$, we note that MatchFAME has around 20% fewer matches remaining compared to all algorithms, but as long as there are enough matches for each edge, one can reliably compute relative rotations and translations for SfM tasks. We believe removing around 20% more matches is not an essential drawback. Furthermore, MatchFAME is faster than the other methods. In conclusion, MatchFAME can achieve a reasonable estimate of matches within a significant short amount of time.

B. Clarifications

We clarify some definitions and expand on various claims mentioned in the paper.

B.1. More on Cycle Consistency and Inconsistency

We referred to a cycle ijk as consistent whenever $\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\leq \boldsymbol{X}_{ik}$, $\boldsymbol{X}_{jk}\boldsymbol{X}_{ki}\leq \boldsymbol{X}_{ji}$ and $\boldsymbol{X}_{ki}\boldsymbol{X}_{ij}\leq \boldsymbol{X}_{kj}$. Note that $\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}$ is a binary matrix with ones whenever there are paths of lengths 2 between keypoints of images i and k and $\boldsymbol{X}_{ik}$ is binary matrix with ones whenever there are paths of lengths 1 (single edges) between keypoints of image i and k. That is, $\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\leq \boldsymbol{X}_{ik}$ means that if keypoints $t_i\in[m_i]$ and $t_k\in[m_k]$ (in images i and k, respectively) are both matched to a keypoint t_j in image j, then they are matched to each other. Therefore, any cycle ijk with corresponding partial permutations $\boldsymbol{X}_{ij}$, $\boldsymbol{X}_{jk}$, $\boldsymbol{X}_{ki}$ is consistent if and only if for any $t_i\in[m_i]$, $t_j\in[m_j]$ and $t_k\in[m_k]$: If two of the events $\boldsymbol{X}_{ij}(t_i,t_j)=1$, $\boldsymbol{X}_{jk}(t_j,t_k)=1$, $\boldsymbol{X}_{ki}(t_k,t_i)=1$ hold true, then the third one holds true as well.

This equivalent reformulation of cycle consistency further clarifies the definition of d_{ijk} in (4). For fixed $\boldsymbol{X}_{ij},\boldsymbol{X}_{jk},\boldsymbol{X}_{ki}\in\mathcal{P}^{l_1,l_2}$, the denominator of the fraction in (4) can be viewed as the number of combinations of three keypoints a, b, c, such that at least two of the three events

$$\boldsymbol{X}_{ij}(a,b)=1,\ \boldsymbol{X}_{jk}(b,c)=1,\text{ and }\boldsymbol{X}_{ki}(c,a)=1 \tag{10}$$

hold. Furthermore, the numerator of the fraction in (4) can be viewed as the total number combinations of three keypoints a, b, c, such that all the three events in (10) hold. Thus, the fraction in (4) indeed measures the level of cycle consistency, and consequently d_{ijk} measures the cycle inconsistency.

We remark that an inequality of two full permutation matrices must be an equality. Therefore, for permutation synchronization the above definition of cycle consistency is equivalent with $\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}=\boldsymbol{X}_{ik}$ (or equivalently, $\boldsymbol{X}_{jk}\boldsymbol{X}_{ki}=\boldsymbol{X}_{ji}$ or $\boldsymbol{X}_{ki}\boldsymbol{X}_{ij}=\boldsymbol{X}_{kj}$ or $\boldsymbol{X}_{ki}\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}=\boldsymbol{I}$). That is, our definition of cycle consistency is a direct extension of the one in group synchronization.

B.2. Cycle-verifiability Helps in Verifying Matches in Cycles

We further interpret the cycle-verifiable condition and clarify its name. We claim that if ijk is a good cycle (w.r.t. ij) ensured by Definition 1 with $a\in I_i$ and $c\in I_j$, then one can verify whether a and c correctly match (i.e., $h(a)=h(c)$) using $b\in I_k$. Indeed, since b matches a and $k\in G_{ij}$, $h(a)=h(b)$. If b and c match then since $k\in G_{ij}$ $h(b)=h(c)$ and consequently $h(a)=h(c)$. Assume on the other hand that b and c do not match. If b matches another point c', then since $k\in G_{ij}$, $h(b)=h(c')\neq h(c)$. If b does not match any point in U_j, then since $k\in G_{ij}$, $h(b)\notin h(U_j)$ (otherwise there exists $c'\in U_j$ such that $h(c')=h(b)$ and since $k\in G_{ij}$ there has to be a match between b and c'.). Since $h(c)\in h(U_j)$ and $h(b)\notin h(U_j)$, $h(b)\neq h(c)$.

C. Proof of Theorem 1

The proof establishes two lemmas, Lemmas 1 and 2, and then uses them to conclude Theorem 1. It is rather technical

Algorithms Dataset			Initial	MatchEig			Spectral			MatchALS			PPM			MatchFAME (ours)		
	n	$\hat{m}$	PR	PR	#M	T	PR	#M	T	PR	#M	T	PR	#M	T	PR	#M	T
Herz-Jesu-P25	25	517	89.6	94.2	73	72	92.2	81	125	93.3	83	9199	92.5	**88**	125	**95.0**	78	**15**
Herz-Jesu-P8	8	386	94.3	95.2	**97**	**1**	95.3	92	4	95.9	76	155	95.4	94	5	**95.9**	83	3
Castle-P30	30	445	71.8	84.7	55	64	80.6	72	99	80.4	76	13583	80.2	**77**	112	**87.9**	61	**15**
Castle-P19	19	314	70.1	79.7	57	23	76.3	**76**	21	77.0	74	1263	77.5	**76**	33	**83.0**	56	**4**
Entry-P10	10	432	75.4	79.9	78	11	82.1	78	30	77.3	77	322	80.7	**83**	34	**83.1**	69	**5**
Fountain-P11	11	374	94.2	95.4	81	8	95.4	93	14	95.7	82	333	95.6	**94**	18	**96.7**	81	**5**

Table 4. Performance on the EPFL datasets. n is the number of cameras; $\hat{m}$, the approximated m, is twice the averaged m_i over $i \in [n]$; PR refers to the precision $|\hat{E} \cap E_g|/|\hat{E}|$, which is expressed in percentage (the higher the better); #M is the ratio (expressed in percentage) between the number of refined matches and the number of initial matches; T is runtime in seconds.

and not so easy to motivate. In order to provide more intuition, we added some clarifying figures.

Convention for figures: In all of these figures, we designate by green lines good keypoint matches, by red lines bad keypoint matches and by dashed orange lines missing keypoint matches. All of these occur between keypoints of two different images. On the other hand, matches between keypoints in an image and universal 3D keypoints are designated by black dotted lines (these correspond to our formal h function). We further color the universal 3D keypoints (in U_{ij}), which represent elements of U_{ij}^{good}, by green. We also color the universal 3D keypoints, which represent elements of U_{ij}^{bad}, in red. In Figure 5, we slightly extend the latter convention and explain it in its caption.

Terminology Review: Recall that $n_\triangle = \text{Tr}(\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\boldsymbol{X}_{ki})$, m is the number of all 3D keypoints, m_{ij} is the number of 3D keypoints that correspond to the 2D keypoints of images i or j and among these, m_{ij}^{bad} is the number of keypoints that match wrong keypoints in the other image, or match no keypoint if a ground-truth match exists. We also denote the number of the rest of points by m_{ij}^{good} (that is, $m_{ij}^{\text{good}} = m_{ij} - m_{ij}^{\text{bad}}$) and recall that these keypoints match the ground-truth keypoints or, do not match any keypoint, if no ground-truth matches exist.

C.1. Upper Bound for the Cycle Inconsistency of Good Edges

This section includes the proof of the following lemma:

Lemma 1. For any $ij \in E_g$, $d_{ijk} \leq m(s_{ik}^* + s_{jk}^*)$.

We remark that in the case of group synchronization, in particular, PS, one can easily show that for any $ij \in E$, $|d_{ijk} - s_{ij}^*| \leq s_{ik}^* + s_{jk}^*$ (see Lemma 1 of [11]). Consequently for $ij \in E_g$, $d_{ijk} \leq s_{ik}^* + s_{jk}^*$. However, in PPS, without the full group structure with a bi-invariant metric, it is harder to prove the weaker bound of Lemma 1. The proof below involves various discrete combinatorial arguments.

Proof. Assume first that $n_\triangle = 0$ and note that (4) implies $d_{ijk} = 1$. Since $d_{ijk} \neq 0$ and $ij \in E_g$, s_{ik}^* and s_{jk}^* cannot be both zero (otherwise this and the fact that $ij \in E_g$ imply that ijk is cycle-consistent and thus $d_{ijk} = 0$). Without loss of generality, assume $s_{jk}^* > 0$. We note that

$$s_{jk}^* = \frac{m_{jk}^{\text{bad}}}{m_{jk}} \geq \frac{1}{m},$$

which implies the desired bound:

$$d_{ijk} = 1 = m \cdot \frac{1}{m} \leq m s_{jk}^* \leq m(s_{ik}^* + s_{jk}^*).$$

Assume next that $n_\triangle > 0$, or equivalently,

$$n_\triangle \geq 1. \tag{11}$$

The next arguments require additional definitions and observations. We recall that any element of I_i represents a 2D keypoint in image i. This keypoint is associated with the index vector (i,j), where $j = 1,\ldots,m_i$ and we can thus view I_i as the set of m_i index vectors. For cycle ijk, $(a,b,c) \in I_i \times I_j \times I_k$ is an (i,jk) tuple if there is a match between a and both b and c (the match can be either good or bad). If (a,b,c) is an (i,jk) tuple and there is no match between b and c, then we refer to (a,b,c) as a bad (i,jk) tuple, otherwise, it is a good (i,jk) tuple. For example, in Figure 5, there are three (i,jk) tuples: $(p_{i,2},p_{j,1},p_{k,1})$, $(p_{i,3},p_{j,3},p_{k,2})$ and $(p_{i,4},p_{j,2},p_{k,3})$. We note that $(p_{i,2},p_{j,1},p_{k,1})$ and $(p_{i,4},p_{j,2},p_{k,3})$ are bad (i,jk) tuples and $(p_{i,3},p_{j,3},p_{k,2})$ is a good (i,jk) tuple. For cycle ijk, we denote by $A_{i,jk}$ the set of (i,jk) tuples in $I_i \times I_j \times I_k$, by $A_{i,jk}^{\text{bad}}$ the set of bad (i,jk) tuples and by $A_{i,jk}^{\text{good}}$ the set of good (i,jk) tuples.

Recall that for a cycle ijk, $n_i = \mathsf{nnz}(\boldsymbol{X}_{ki}\boldsymbol{X}_{ij})$, n_j and n_k are analogously defined, and $n_\triangle = \text{Tr}(\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\boldsymbol{X}_{ki})$. Also recall the notation $p_{i,j}$ for elements of I_i. We note that (a,b,c) is an (i,jk) tuple if and only if (iff) $a = p_{i,u}$, $b = p_{j,v}$, $c = p_{k,w}$ and both $\boldsymbol{X}_{ki}(w,u) = 1$ and $\boldsymbol{X}_{ij}(u,v) = 1$ (so $p_{i,u}$ matches to both $p_{j,v}$ and $p_{k,w}$). We further note that the latter two requirements are equivalent with $\boldsymbol{X}_{ki}\boldsymbol{X}_{ij}(w,v) = 1$. Indeed, since $\boldsymbol{X}_{ki}\boldsymbol{X}_{ij}(w,v) = \sum_{u'\in[m_i]} \boldsymbol{X}_{ki}(w,u')\boldsymbol{X}_{ij}(u',v)$ and $\boldsymbol{X}_{ki}\boldsymbol{X}_{ij}$ is a partial permutation, then $\boldsymbol{X}_{ki}\boldsymbol{X}_{ij}(w,v) = 1$ iff $\boldsymbol{X}_{ki}(w,u) = \boldsymbol{X}_{ij}(u,v) = 1$ for some $u \in [m_i]$. Therefore

$$|A_{i,jk}| = n_i.$$

By the same way we note that

$$|A_{j,ki}| = n_j \text{ and } |A_{k,ij}| = n_k.$$

Similarly, note that (a,b,c) is a good (i,jk) tuple iff $a = p_{i,u}$, $b = p_{j,v}$, $c = p_{k,w}$, $\boldsymbol{X}_{ki}(w,u) = 1$, $\boldsymbol{X}_{ij}(u,v) = 1$ and $\boldsymbol{X}_{jk}(v,w) = 1$. The latter three requirements are equivalent with $\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\boldsymbol{X}_{ki}(u,u) = 1$. Indeed, the following equation

$$\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\boldsymbol{X}_{ki}(u,u) = \sum_{v'\in[m_j], w'\in[m_k]} \boldsymbol{X}_{ij}(u,v')\boldsymbol{X}_{jk}(v',w')\boldsymbol{X}_{ki}(w',u)$$

and the fact that $\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\boldsymbol{X}_{ki}$ is a partial permutation imply this equivalence. Therefore,

$$|A^{\text{good}}_{i,jk}| = n_\Delta \text{ and } |A^{\text{bad}}_{i,jk}| = n_i - n_\Delta.$$

Similarly, we conclude that

$$|A^{\text{bad}}_{j,ki}| = n_j - n_\Delta \text{ and } |A^{\text{bad}}_{k,ij}| = n_k - n_\Delta.$$

Using these observations and (4), one can rewrite d_{ijk} as follows

$$d_{ijk} = \frac{|A^{\text{bad}}_{i,jk}| + |A^{\text{bad}}_{j,ki}| + |A^{\text{bad}}_{k,ij}|}{|A^{\text{bad}}_{i,jk}| + |A^{\text{bad}}_{j,ki}| + |A^{\text{bad}}_{k,ij}| + 3n_\Delta}. \tag{12}$$

Let us assume that $(a,b,c) \in A^{\text{bad}}_{i,jk}$ and show that

$$h(c) \in U^{\text{bad}}_{ik} \cup U^{\text{bad}}_{jk}. \tag{13}$$

The assumption $ij \in E_g$ implies that there is a good match between a and b.

We claim that if there is also a good match between a and c, then $h(b) = h(a) = h(c) \in U_{ij} \cap U_{ik} \cap U_{jk}$. Indeed, assume $a = p_{i,u}$, $b = p_{j,v}$, $c = p_{k,w}$ and $h(a) = l$, i.e., $\boldsymbol{P}^*_i(u,l) = 1$. Because there exists a good match between a and b, $\boldsymbol{X}^*_{ij}(u,v) = \boldsymbol{X}_{ij}(u,v) = 1$. Since $\boldsymbol{X}^*_{ij}(u,v) = \boldsymbol{P}^*_i\boldsymbol{P}^{*T}_j(u,v) = \sum_{d\in[m]} \boldsymbol{P}^*_i(u,d)\boldsymbol{P}^*_j(v,d)$ and $\boldsymbol{P}^*_j$ is a partial permutation, $\boldsymbol{P}^*_j(v,l) = 1$ and thus $h(b) = l$. Similarly, since there exists a good match between a and c, $\boldsymbol{P}^*_k(w,l) = 1$ and $h(c) = l$. Therefore $h(a) = h(b) = h(c) = l \in U_{ij} \cap U_{ik} \cap U_{jk}$.

Since $(a,b,c) \in A^{\text{bad}}_{i,jk}$, there is no match between b and c and thus $h(c) \in U^{\text{bad}}_{jk}$, which implies (13).

If on the other hand, there is a bad match between a and c, then $h(c) \in U^{\text{bad}}_{ik}$, which also implies (13).

In view of (13), the function $f(a,b,c) = h(c)$ maps $A^{\text{bad}}_{i,jk}$ to $U^{\text{bad}}_{ik} \cup U^{\text{bad}}_{jk}$. We note that this function is injective. Indeed, since $\boldsymbol{X}_{ik}$ and $\boldsymbol{X}_{jk}$ are partial permutations, for any

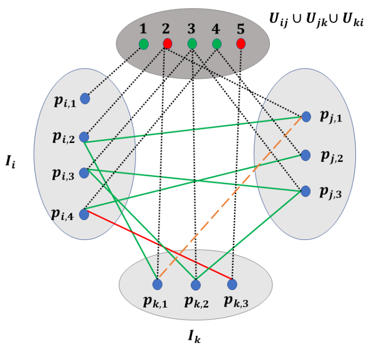

Figure 5. A demonstration for clarifying the definition of (i,jk) tuples, good (i,jk) tuples, bad (i,jk) tuples as well as the function f. The actual use of the figure is clarified when it is referred to. Unlike other figures (that focus on points in U_{ij} and not in $U_{ij} \cup U_{jk} \cup U_{ki}$), the red dots correspond to keypoints in $U^{\text{bad}}_{ij} \cup U^{\text{bad}}_{jk} \cup U^{\text{bad}}_{ik}$ and the green dots correspond to the rest of keypoints in $U_{ij} \cup U_{jk} \cup U_{ki}$. We note that $U^{\text{bad}}_{ik} = \{5\}$, $U^{\text{bad}}_{jk} = \{2\}$ and $U^{\text{bad}}_{ij} = \varnothing$.

$h(c) \in U^{\text{bad}}_{ik} \cup U^{\text{bad}}_{jk}$, where $c \in I_k$, if there are a and b in I_i and I_j, respectively, such that there are matches between them and c and $f(a,b,c) = h(c)$, then a and b are unique. Figure 5 demonstrates f and its proven injectivity in a special case. In this case, $A^{\text{bad}}_{i,jk} = \{(p_{i,4}, p_{j,2}, p_{k,3}), (p_{i,2}, p_{j,1}, p_{k,1})\}$, $f(p_{i,4}, p_{j,2}, p_{k,3}) = h(p_{k,3}) = 5 \in U^{\text{bad}}_{ik}$ and $f(p_{i,2}, p_{j,1}, p_{k,1}) = h(p_{k,1}) = 2 \in U^{\text{bad}}_{jk}$.

By the cardinality property of an injective map,

$$|A^{\text{bad}}_{i,jk}| \le |U^{\text{bad}}_{ik} \cup U^{\text{bad}}_{jk}| \le |U^{\text{bad}}_{ik}| + |U^{\text{bad}}_{jk}| = m^{\text{bad}}_{ik} + m^{\text{bad}}_{jk}.$$

Similarly, the same bound holds for $|A^{\text{bad}}_{j,ki}|$ and $|A^{\text{bad}}_{k,ij}|$ and consequently,

$$|A^{\text{bad}}_{i,jk}| + |A^{\text{bad}}_{j,ki}| + |A^{\text{bad}}_{k,ij}| \le 3(m^{\text{bad}}_{ik} + m^{\text{bad}}_{jk}). \tag{14}$$

Therefore the combination of (11) and (14) with the definition of s^*_{ik} as well as s^*_{jk} (i.e., noting that $s^*_{ik} = m^{\text{bad}}_{ik}/m_{ik} = 1 - m^{\text{good}}_{ik}/m_{ik}$ and $s^*_{jk} = m^{\text{bad}}_{jk}/m_{jk} = 1 - m^{\text{good}}_{jk}/m_{jk}$) yields

$$\begin{aligned}
d_{ijk} &= \frac{|A^{\text{bad}}_{i,jk}| + |A^{\text{bad}}_{j,ki}| + |A^{\text{bad}}_{k,ij}|}{|A^{\text{bad}}_{i,jk}| + |A^{\text{bad}}_{j,ki}| + |A^{\text{bad}}_{k,ij}| + 3n_\Delta} \\
&\le \frac{3(m^{\text{bad}}_{ik} + m^{\text{bad}}_{jk})}{3(m^{\text{bad}}_{ik} + m^{\text{bad}}_{jk}) + 3} \\
&\le \frac{3m^{\text{bad}}_{ik}}{3m^{\text{bad}}_{ik} + 3} + \frac{3m^{\text{bad}}_{jk}}{3m^{\text{bad}}_{jk} + 3} \\
&= \frac{3s^*_{ik}}{3s^*_{ik} + \frac{3}{m_{ik}}} + \frac{3s^*_{jk}}{3s^*_{jk} + \frac{3}{m_{jk}}} \\
&\le m_{ik}s^*_{ik} + m_{jk}s^*_{jk} \\
&\le m(s^*_{ik} + s^*_{jk}).
\end{aligned}$$

□

C.2. Lower Bound for the Averaged Cycle Inconsistency Among Good Cycles

This section includes the proof of the following lemma:

Lemma 2. If $G=(V,E)$ is p_v-cycle verifiable, then

$$\frac{1}{3}p_v s_{ij}^* \leq \frac{1}{|G_{ij}|}\sum_{k\in G_{ij}} d_{ijk} \quad \forall ij\in E. \tag{15}$$

Proof. We assume several cases.
Case I: $ij\in E_g$. The left hand side of (15) is zero and its right hand side (RHS) is also zero since for any $ij\in E_g$ and $k\in G_{ij}$, $d_{ijk}=0$.
Case II: $ij\in E_b$ and $k\in G_{ij}$. Denote

$$N_{ijk}^{\text{bad}} = |A_{i,jk}^{\text{bad}}|+|A_{j,ki}^{\text{bad}}|+|A_{k,ij}^{\text{bad}}|,$$

and note that in view of (12)

$$\sum_{k\in G_{ij}} d_{ijk} = \sum_{k\in G_{ij}} \frac{N_{ijk}^{\text{bad}}}{N_{ijk}^{\text{bad}}+3n_\Delta}. \tag{16}$$

We thus need to lower bound the RHS of (16) in order to conclude (15).

We first derive the bound $n_\Delta \leq m_{ij}^{\text{good}}$. Figure 6 demonstrates the definitions below and the desired bound in a very special case. Let D_{ijk} denote the set of indices of diagonal entries of $\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\boldsymbol{X}_{ki}$ that equal 1. Note that $|D_{ijk}|=n_\Delta$ due to the fact that $n_\Delta = \text{Tr}(\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\boldsymbol{X}_{ki})$. Also, since $\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\boldsymbol{X}_{ki}$ is of size $m_i\times m_i$, $D_{ijk}\subseteq [m_i]$. We can thus assign for $d\in D_{ijk}$, $p_{i,d}=a\in I_i$. Because $\boldsymbol{X}_{ij}\boldsymbol{X}_{jk}\boldsymbol{X}_{ki}(d,d)=1$, there exists $1\leq u\leq m_j$ and $1\leq v\leq m_k$ such that $\boldsymbol{X}_{ij}(d,u)=\boldsymbol{X}_{jk}(u,v)=\boldsymbol{X}_{ki}(v,d)=1$. Therefore, we note that for $b:=p_{j,u}\in I_j$ and $c:=p_{k,v}\in I_k$, there exist matches between a and b, b and c, as well as c and a. Since $k\in G_{ij}$, $jk\in E_g$ and $ki\in E_g$ and thus $h(b)=h(c)=h(a)$ (see the same argument in the paragraph below (13), where it is enough to just assume that either $jk\in E_g$ or $ki\in E_g$); we denote the latter common value by l. Therefore $\boldsymbol{P}_i^*(a,l)=\boldsymbol{P}_j^*(b,l)=\boldsymbol{X}_{ij}(a,b)=1$. By definition of U_{ij}^{good}, $l\in U_{ij}^{\text{good}}$. Let f_{ijk} be a function from D_{ijk} to U_{ij}^{good} such that $f_{ijk}(d)=l$. We note that it is injective since for any $d\neq d'\in D_{ijk}$, $p_{i,d}\neq p_{i,d'}$, therefore $h(p_{i,d})\neq h(p_{i,d'})$. By the cardinality property of an injective map, $n_\Delta\leq |U_{ij}^{\text{good}}|=m_{ij}^{\text{good}}$.

Next, We prove an upper bound of N_{ijk}^{bad}. We assume without loss of generality that $(a,b,c)\in A_{i,jk}^{\text{bad}}$. Since $ik,jk\in E_g$, the matches from a to c and from b to c are correct. Therefore, the match from a to b is wrong and $h(a)\in U_{ij}^{\text{bad}}$. Denote by $g_i: A_{i,jk}^{\text{bad}}\to U_{ij}^{\text{bad}}$ the function which maps $(a,b,c)\in A_{i,jk}^{\text{bad}}$ to $h(a)\in U_{ij}^{\text{bad}}$. Figure 7 illustrates g_i in a special case. This function is injective since for any $x\in U_{ij}^{\text{bad}}$, $g_i^{-1}(x)$ contains at most one element $(a,b,c)\in A_{i,jk}^{\text{bad}}$. Indeed, if $g_i(a,b,c)=x$, then there must exist $a\in I_i$ such that $h(a)=x$, c in I_k such that there is a match between a and c, and b in I_j such that there is a match between a and b (and no match between b and c). Note that there is a match between at most one keypoint in I_k and a and thus there is at most one such c. Similarly, there is at most one such b. Since there is at most one keypoint in I_i which corresponds to the 3D keypoint x, there is at most one such a. The injectivity of g_i implies $|A_{i,jk}^{\text{bad}}|\leq|U_{ij}^{\text{bad}}|=m_{ij}^{\text{bad}}$. Similarly, $|A_{j,ki}^{\text{bad}}|\leq m_{ij}^{\text{bad}}$ and $|A_{k,ij}^{\text{bad}}|\leq m_{ij}^{\text{bad}}$. Thus, for any $k\in G_{ij}$

$$0\leq N_{ijk}^{\text{bad}}\leq 3m_{ij}^{\text{bad}}. \tag{17}$$

Next, we establish a lower bound of N_{ijk}^{bad}. For this purpose, we construct an injective map $g_{ij,k}$ from $U_{ij}^{\text{bad}}\cap h(I_k)$ to $A_{i,jk}^{\text{bad}}\cup A_{j,ki}^{\text{bad}}\cup A_{k,ij}^{\text{bad}}$. It will allow us to lower bound $N_{ijk}^{\text{bad}}=|A_{i,jk}^{\text{bad}}|+|A_{j,ki}^{\text{bad}}|+|A_{k,ij}^{\text{bad}}|$ by the cardinality of

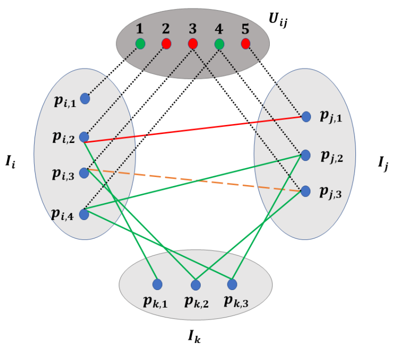

Figure 6. An illustration of n_Δ, D_{ijk}, f_{ijk} and m_{ij}^{good}. Note that n_Δ is equal to the number of green triangles with vertices in I_i,I_j,I_k. Since there is only one such triangle, $n_\Delta=1$. This triangle (with keypoints $p_{i,4}$,$p_{j,2}$,$p_{k,3}$) is associated with the keypoint in I_i with index 4 and thus $D_{ijk}=\{4\}$. Since it is also associated with the universal keypoint with index 4, the function f_{ijk} maps 4 to $4\in U_{ij}^{\text{good}}$. At last, note that $m_{ij}^{\text{good}}=|U_{ij}^{\text{good}}|=2$.

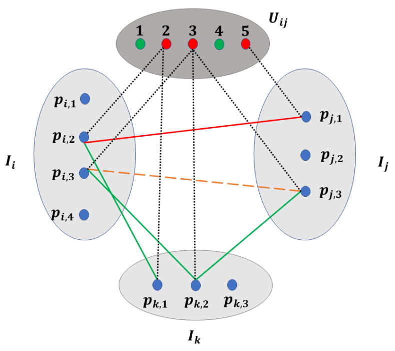

Figure 7. An illustration of g_i, $g_{ij,k}$ and the injectivity of both functions in a very special case. In this example, the only element of $A_{i,jk}^{\text{bad}}$ is $(p_{i,2},p_{j,1},p_{k,1})$. Since we defined $g_i(a,b,c)=h(a)$, we obtain that $g_i(p_{i,2},p_{j,1},p_{k,1})=h(p_{i,2})=2\in U_{ij}^{\text{bad}}$. Note that $U_{ij}^{\text{bad}}\cap h(I_k)=\{2,3\}$. Recall that the function $g_{ij,k}$ maps $x\in U_{ij}^{\text{bad}}\cap h(I_k)$ to the bad (j,ki) tuple or bad (k,ij) tuple that involves $c\in I_k$ with $h(c)=x$. In this example, the function $g_{ij,k}$ maps $2\in U_{ij}$ to the bad (i,jk) tuple $(p_{i,2},p_{j,1},p_{k,1})$. It maps $3\in U_{ij}$ to the bad (k,ij) tuple $(p_{k,2},p_{i,3},p_{j,3})$.

$U_{ij}^{\text{bad}} \cap h(I_k)$. Note that $U_{ij}^{\text{bad}} \subseteq U_{ij} = h(I_i) \cup h(I_j)$. Therefore any element of $U_{ij}^{\text{bad}} \cap h(I_k)$ is either in $h(I_i)$ or $h(I_j) \setminus h(I_i)$.

In the case where $x \in U_{ij}^{\text{bad}} \cap h(I_k)$ and $x \in h(I_i)$, we will show that there exist either $(a,b,c) \in A_{i,jk}^{\text{bad}}$ or $(c,a,b) \in A_{k,ij}^{\text{bad}}$ such that $h(c) = x$. In the case where $x \in U_{ij}^{\text{bad}} \cap h(I_k)$ and $x \in h(I_j) \setminus h(I_i)$, then one can similarly show that there exists either $(b,c,a) \in A_{j,ki}^{\text{bad}}$ or $(c,a,b) \in A_{k,ij}^{\text{bad}}$ such that $h(c) = x$. These arguments induce a map $g_{ij,k}$ from $U_{ij}^{\text{bad}} \cap h(I_k)$ to $A_{i,jk}^{\text{bad}} \cup A_{j,ki}^{\text{bad}} \cup A_{k,ij}^{\text{bad}}$ which maps x to its corresponding bad tuple. Since $h(c) = x$, $g_{ij,k}$ is injective. Figure 7 illustrates $g_{ij,k}$ in a special case.

We thus assume that $x \in U_{ij}^{\text{bad}} \cap h(I_k)$ and $x \in h(I_i)$. Note that the latter requirement implies the existence of $a \in I_i$ such that $h(a) = x$. Since $x \in h(I_k)$, there exists $c \in I_k$ such that $h(c) = x$ and since $ik \in E_g$ there is a good match between a and c. Note that there cannot be a good match between a and any $b \in I_j$, otherwise $x \notin U_{ij}^{\text{bad}}$. Therefore, there are two cases to consider. In the first case there exists $b \in I_j$ such that there is a wrong match between a and b. This implies that $h(b) \neq h(a)$ and since we showed above that $h(a) = h(c)$, we conclude that $h(b) \neq h(c)$. The latter observation and the fact that $jk \in E_g$ imply that there is no match between b and c and thus (a,b,c) is a bad (i,jk) tuple, that is, $(a,b,c) \in A_{i,jk}^{\text{bad}}$. In the second case, there exists $b \in I_j$ such that $h(b) = h(a)$, but there is no match between b and a (the previous case considered the scenario where there exists $b \in I_j$ such that a and b match; furthermore, if $h(a) \neq h(b)$ for all $b \in I_j$, then $x \in U_{ij}^{\text{good}}$). Since $h(a) = h(c) = h(b)$ and $jk \in E_g$, there is a match between b and c. Therefore, (c,a,b) is a bad (k,ij) tuple, that is, $(c,a,b) \in A_{k,ij}^{\text{bad}}$. Following the above ideas, this concludes the injectivity of $g_{ij,k}$. This injectivity implies

$$\sum_{x \in U_{ij}^{\text{bad}}} 1_{\{x \in h(I_k)\}} = |U_{ij}^{\text{bad}} \cap h(I_k)| \leq |A_{i,jk}^{\text{bad}} \cup A_{j,ki}^{\text{bad}} \cup A_{k,ij}^{\text{bad}}| \leq |A_{i,jk}^{\text{bad}}| + |A_{j,ki}^{\text{bad}}| + |A_{k,ij}^{\text{bad}}| = N_{ijk}^{\text{bad}}. \quad (18)$$

In order to apply (18) we lower bound a certain sum of $1_{\{x \in h(I_k)\}}$. Our argument assumes that $x \in U_{ij}^{\text{bad}}$. Since $x \in U_{ij}$, we conclude WLOG that $x \in h(I_i)$. Therefore there exists $a \in I_i$ such that $h(a) = x$. By the p_v-cycle verifiability condition, a is verifiable w.r.t. ij in at least $p_v|G_{ij}|$ good cycles. For any such cycle ijk that a is verifiable in, let $b \in I_k$ match a (for convenience, we demonstrate x, a and b in Figure 8). Since $ik \in E_g$, the match between a and b is a good match and thus $x = h(a) = h(b)$. Since $b \in I_k$, $h(b) \in h(I_k)$ and thus $x \in h(I_k)$. That is, we have proved that if $k \in G_{ij}$ and a is verifiable in ijk, then $x \in h(I_k)$. We have at least $p_v|G_{ij}|$ such k's and thus

$$\sum_{k \in G_{ij}} 1_{\{x \in h(I_k)\}} \geq p_v|G_{ij}| \quad \text{for any } x \in U_{ij}. \quad (19)$$

We combine the above two inequalities as follows. Summing both sides of (18) over $k \in G_{ij}$, exchanging the order of summation and applying (19) result in

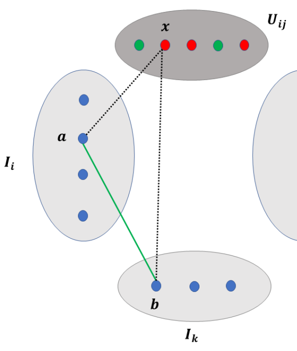

Figure 8. Visual demonstration of keypoints involved in the argument for bounding $\sum_{k \in G_{ij}} 1_{\{x \in h(I_k)\}}$.

$$p_v|G_{ij}|m_{ij}^{\text{bad}} = p_v|G_{ij}||U_{ij}^{\text{bad}}| \leq \sum_{k \in G_{ij}} N_{ijk}^{\text{bad}}. \quad (20)$$

Using the above bound we will bound from below

$$\sum_{k \in G_{ij}} \frac{N_{ijk}^{\text{bad}}}{N_{ijk}^{\text{bad}} + 3m_{ij}^{\text{good}}}$$

and we will then use (16) to conclude the desired inequality. We denote

$$F(x) = \frac{x}{x + \gamma} \quad \text{where } \gamma = 3m_{ij}^{\text{good}}. \quad (21)$$

Note that $F(0) = 0$,

$$F(3m_{ij}^{\text{bad}}) = \frac{3m_{ij}^{\text{bad}}}{3m_{ij}^{\text{bad}} + 3m_{ij}^{\text{good}}} = s_{ij}^*$$

and $F(x)$ is concave. Applying the definition of F, Jensen's inequality, (20) and (21) yield

$$\begin{aligned}
\sum_{k \in G_{ij}} \frac{N_{ijk}^{\text{bad}}}{N_{ijk}^{\text{bad}} + 3m_{ij}^{\text{good}}} &= \sum_{k \in G_{ij}} F(N_{ijk}^{\text{bad}}) \\
&\geq \sum_{k \in G_{ij}} \left(\left(1 - \frac{N_{ijk}^{\text{bad}}}{3m_{ij}^{\text{bad}}}\right) F(0) + \frac{N_{ijk}^{\text{bad}}}{3m_{ij}^{\text{bad}}} F(3m_{ij}^{\text{bad}}) \right) \\
&= \sum_{k \in G_{ij}} \frac{N_{ijk}^{\text{bad}}}{3m_{ij}^{\text{bad}}} F(3m_{ij}^{\text{bad}}) \\
&\geq \frac{1}{3} p_v |G_{ij}| F(3m_{ij}^{\text{bad}}) = \frac{1}{3} p_v |G_{ij}| s_{ij}^*.
\end{aligned}$$

The combination of this inequality with (16) concludes the proof of the lemma. □

C.3. Conclusion of Theorem 1

We prove the main theorem by induction, using Lemmas 1 and 2. For $t=0$, the definition of $s_{ij}^{(0)}$, Lemma 2 and the definition of λ imply that for all $ij \in E$:

$$s_{ij}^{(0)} = \frac{\sum_{k\in N_{ij}} d_{ijk}}{|N_{ij}|} \geq \frac{\sum_{k\in G_{ij}} d_{ijk}}{|N_{ij}|} \geq \frac{p_v}{3}\frac{|G_{ij}|}{|N_{ij}|} s_{ij}^* \geq \frac{p_v}{3}(1-\lambda)s_{ij}^*.$$

We further note by using again the above definitions and the fact that for all $ij \in E$ $0 \leq d_{ijk} \leq 1$ that for $ij \in E_g$,

$$s_{ij}^{(0)} = \frac{\sum_{k\in N_{ij}} d_{ijk}}{|N_{ij}|} = \frac{\sum_{k\in B_{ij}} d_{ijk}}{|N_{ij}|} \leq \frac{\sum_{k\in B_{ij}} 1}{|N_{ij}|} \leq \lambda \leq \frac{1}{2\beta_0}.$$

Therefore, the theorem is proved when $t=0$.

Next, we assume that the theorem holds for iterations $0,1,\cdots,t$ and show that it also holds for iteration $t+1$. Applying the definition of $s_{ij}^{(t+1)}$, the positivity of the terms in the sum, the induction assumption $\frac{1}{2\beta_t} \geq \max_{ij\in E_g} s_{ij}^{(t)}$, Lemma 2 and the definition of λ, we obtain for any $ij \in E_b$

$$\begin{aligned} s_{ij}^{(t+1)} &= \frac{\sum_{k\in N_{ij}} e^{-\beta_t(s_{ik}^{(t)}+s_{jk}^{(t)})} d_{ijk}}{\sum_{k\in N_{ij}} e^{-\beta_t(s_{ik}^{(t)}+s_{jk}^{(t)})}} \\ &\geq \frac{\sum_{k\in G_{ij}} e^{-\beta_t(s_{ik}^{(t)}+s_{jk}^{(t)})} d_{ijk}}{\sum_{k\in N_{ij}} e^{-\beta_t(s_{ik}^{(t)}+s_{jk}^{(t)})}} \\ &\geq \frac{\sum_{k\in G_{ij}} e^{-1} d_{ijk}}{|N_{ij}|} \\ &\geq \frac{p_v}{3e}\frac{|G_{ij}|}{|N_{ij}|} s_{ij}^* \\ &\geq \frac{(1-\lambda)p_v}{3e} s_{ij}^*. \end{aligned} \tag{22}$$

Note that $xe^{-\alpha x} \leq \frac{1}{\alpha e}$ for any $\alpha > 0$ and $x \geq 0$. In particular, for $\alpha = \beta_t(1-\lambda)p_v/3e$ and $x = s_{ik}^* + s_{jk}^*$,

$$e^{-\beta_t(s_{ik}^*+s_{jk}^*)(1-\lambda)p_v/3e}(s_{ik}^*+s_{jk}^*) \leq \frac{3}{\beta_t(1-\lambda)p_v}. \tag{23}$$

Applying the definition of $s_{ij}^{(t+1)}$, the fact that $d_{ijk}=0$ for any $ij \in E_g$ and $k \in G_{ij}$, Lemma 1, the induction assumption $s_{ij}^{(t)} \geq \frac{(1-\lambda)p_v}{3e} s_{ij}^*$ for $ij \in E_b$ (for the numerator) and the positivity of the relevant terms (for the denominator), the induction assumption $s_{ij}^{(t)} \leq 1/(2\beta_t)$ for all $ij \in E_g$, (23) and the definition of λ, we obtain for all $ij \in E_g$

$$\begin{aligned} s_{ij}^{(t+1)} &= \frac{\sum_{k\in N_{ij}} e^{-\beta_t(s_{ik}^{(t)}+s_{jk}^{(t)})} d_{ijk}}{\sum_{k\in N_{ij}} e^{-\beta_t(s_{ik}^{(t)}+s_{jk}^{(t)})}} \\ &= \frac{\sum_{k\in B_{ij}} e^{-\beta_t(s_{ik}^{(t)}+s_{jk}^{(t)})} d_{ijk}}{\sum_{k\in N_{ij}} e^{-\beta_t(s_{ik}^{(t)}+s_{jk}^{(t)})}} \\ &\leq \frac{\sum_{k\in B_{ij}} e^{-\beta_t(s_{ik}^{(t)}+s_{jk}^{(t)})} m(s_{ik}^*+s_{jk}^*)}{\sum_{k\in N_{ij}} e^{-\beta_t(s_{ik}^{(t)}+s_{jk}^{(t)})}} \\ &\leq \frac{\sum_{k\in B_{ij}} e^{-\beta_t(s_{ik}^*+s_{jk}^*)\frac{(1-\lambda)p_v}{3e}} m(s_{ik}^*+s_{jk}^*)}{\sum_{k\in G_{ij}} e^{-\beta_t(s_{ik}^{(t)}+s_{jk}^{(t)})}} \\ &\leq \frac{m\sum_{k\in B_{ij}} e^{-\beta_t(s_{ik}^*+s_{jk}^*)\frac{(1-\lambda)p_v}{3e}} (s_{ik}^*+s_{jk}^*)}{|G_{ij}|e^{-1}} \\ &\leq \frac{m\sum_{k\in B_{ij}} \frac{3}{\beta_t(1-\lambda)p_v}}{|G_{ij}|e^{-1}} \\ &= \frac{3m|B_{ij}|}{|G_{ij}|e^{-1}} \cdot \frac{1}{\beta_t(1-\lambda)p_v} \\ &\leq \frac{6em\lambda}{(1-\lambda)^2 p_v} \cdot \frac{1}{2\beta_t}. \end{aligned}$$

We note that the assumption $\lambda < 1 + \frac{3em}{p_v} - \sqrt{\frac{3em}{p_v}(2+\frac{3em}{p_v})}$ is equivalent with $\frac{6em\lambda}{(1-\lambda)^2 p_v} < 1$. Therefore by taking $\beta_{t+1} = r\beta_t$ with $1 < r < \frac{(1-\lambda)^2 p_v}{6em\lambda}$, we guarantee that for any $ij \in E_g$, $s_{ij}^{(t+1)} \leq \frac{1}{2\beta_{t+1}}$, that is, $\max_{ij\in E_g} s_{ij}^{(t+1)} \leq \frac{1}{2\beta_{t+1}} = \frac{1}{2\beta_0 r^t}$. This implication and (22) conclude the proof of the theorem.

D. Discussion of a Possible Theoretical Extension

Although our current analysis assumes no noise on the set of good edges, one can relax this assumption. Indeed, one can assume sufficiently small noise on good edges so that for all cycles ijk and a sufficiently small positive constant δ: $|d'_{ijk} - d_{ijk}| < \delta$, where d_{ijk} and d'_{ijk} are respectively the cycle inconsistencies with and without noise on good edges. Using a basic perturbation analysis, similarly as in the proof of Theorem 1, with a carefully chosen set of the reweighting parameters β_t, one can prove approximate separation of good and bad edges. In particular, the maximum value of the estimated s_{ij} on good edges is proportional to δ. Removing the bad edges (with estimated s_{ij} larger than this threshold), one can then approximately solve the PPS problem with a subsequent spectral solver. An approximate recovery theorem for the absolute partial permutations using the filtered edges can be established using spectral graph theory.