Scalable Cluster-Consistency Statistics for Robust Multi-Object Matching

Yunpeng Shi* Shaohan Li† Tyler Maunu‡ Gilad Lerman†
*Program in Applied and Computational Mathematics, Princeton University
†School of Mathematics, University of Minnesota
‡Department of Mathematics, Brandeis University

2021 International Conference on 3D Vision (3DV) | 978-1-6654-2688-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/3DV53792.2021.00045

Abstract

We develop new statistics for robustly filtering corrupted keypoint matches in the structure from motion pipeline. The statistics are based on consistency constraints that arise within the clustered structure of the graph of keypoint matches. The statistics are designed to give smaller values to corrupted matches and than uncorrupted matches. These new statistics are combined with an iterative reweighting scheme to filter keypoints, which can then be fed into any standard structure from motion pipeline. This filtering method can be efficiently implemented and scaled to massive datasets as it only requires sparse matrix multiplication. We demonstrate the efficacy of this method on synthetic and real structure from motion datasets and show that it achieves state-of-the-art accuracy and speed in these tasks.

1. Introduction

The problem of matching multiple objects, namely, multi-object matching, arises in different computer vision applications, such as 3D shape matching [10] and structure from motion (SfM) [18]. In SfM and other 3D reconstruction problems, each object is an image that contains a set of keypoints that are generated and matched using various automatic procedures. In practice, due to differing viewpoints, every pair of images only contains a partial set of shared keypoints among all possible keypoints, and the estimated matches are typically corrupted. Common sources of corruption of the estimated keypoint matches are scene occlusion, change of illumination, viewing distance and perspective, repetitive patterns and ambiguous symmetry [30].

Formally, the general problem of multi-object matching in SfM aims to assign absolute matchings between keypoints and their corresponding 3D scene points. However, in SfM and 3D reconstruction problems, one only needs to estimate the fundamental matrices of each image given the corrupted partial matches [18]. Therefore, for such applications, it is sufficient to estimate only a subset of the ground-truth keypoint matches without estimating all of the absolute matches, since it only takes 7 matches to specify the fundamental matrix [7].

The common matching procedure in SfM directly compares SIFT [15] feature descriptors and refines them by bidirectional matching [7]. Typically, some classical methods are used to make the SfM pipeline robust to keypoint mismatches. For example, it is common to use either the least median of squares or RANSAC for fundamental matrix estimation. Bundle adjustment, the final step in the SfM pipeline, can also be used to correct keypoint matches.

Different methods have been proposed for solving the multi-object matching in its broad sense, but they have been either unscalable or practically ineffective for SfM. Nevertheless, it is interesting to note mathematical ideas that were used before. One basic mathematical framework that applies to the matching of only two objects is that of the quadratic assignment problem [14]. Another mathematical framework for multi-object matching with complete matching (and thus does not apply to partial matching) is Permutation Synchronization (PS) [19]. These two formulations are not directly applicable to SfM and are hard to compute [21, 6, 19]. A more natural formulation for the matching problem in SfM is Partial Permutation Synchronization (PPS) [4]. It aims to address the setting of partial matching, where some keypoints in one image may not match those in another image. Unlike permutation synchronization, PPS is not a special case of the general problem of group synchronization [13, 20], and thus there are no clean and universal approaches to address it. Its existing solutions [4, 31] are computationally demanding, so they may not be effectively used within the SfM pipeline.

Some recent attempts have tried to leverage the consistency structure of the graph of keypoints (which is different from the common graph used for the PPS problem, whose nodes represent images) [8, 24]. In particular, [8] uses such ideas for a distributed implementation of [31]; however, it is not scalable when run on a single machine (it is typically only 10 times faster than [31]). Furthermore, [24] utilizes QuickMatch [28] on the graph of keypoints to robustly find keypoint matches. However, [28] clusters the keypoints through a for loop over all keypoint matches; thus matching errors may accumulate along this sequential and heuristic procedure. Both [24] and [28] also require additional keypoint features, so they are not standard PPS algorithms that are purely based on cycle-consistency information in the keypoint graph. Despite these original and innovative directions, we believe there is still more room to quantitatively explore the structure of this graph and accelerate the standard PPS algorithms to handle large-scale SfM matching data.

The goal of this work is to propose a rather simple and fast method to solve a weaker formulation of the partial permutation

978-1-6654-2688-6/21/$31.00 ©2021 IEEE
DOI 10.1109/3DV53792.2021.00045

synchronization problem under high corruption. The strategies we propose do not seek to capture all correct keypoint matches but rather find a good, consistent subset of them. Specifically, we seek the intersection of the good matches and the given matches. Most importantly, this method can be effectively used within the SfM pipeline, since one does not need all keypoint matches to estimate downstream quantities, like the fundamental matrix. Our method is motivated by some mathematical insights that have not been fully leveraged in the existing literature.

1.1. Previous Works

Several methods were proposed for solving very special instances of multi-object matching by permutation synchronization [19, 11, 25]; these instances are inapplicable to SfM. PPS is more relevant for the general setting of multi-object matching, although it is applied to SfM in a cumbersome way: The estimated absolute partial permutations are used to estimate the ground-truth relative partial permutations and consequently the ground-truth keypoint matches. In order to solve PPS, several works extended convex relaxation methods for permutation synchronization by relaxing permutation matrices to doubly stochastic matrices [4, 31, 12]. However, these methods are often inaccurate, unrobust to corruption, slow and not scalable for practical instances of SfM. Nonconvex methods for PPS include [16, 1, 29, 2]. The spectral method [19] and the projected power method (PPM) [11, 3] for permutation synchronization can be extended to PPS, but they are not sufficiently accurate and scalable and also require an estimate of the unknown number of total unique keypoints. Other recent works begin to examine consistency constraints of the underlying keypoint graph [8, 24] and we discussed them above. Among the aforementioned PPS algorithms, the two fastest ones are [16] and [19]. However, these and other spectral-based PS/PPS methods [3, 11] require computation of the top m eigenvectors, where m is the number of unique keypoints, which is also called the universe size. These eigenvectors form a dense matrix of size $N \times m$ where N is the total number of keypoints in all images. For large SfM problems, N and m can be of order 10^6 and 10^5, respectively, so the memory requirement is >100 GB memory and cannot be addressed by a common personal computer. Thus, in order to handle the large-scale SfM data, it is crucial to only involve sparse matrix operations in PPS algorithms, which has not been considered in previous works.

1.2. This Work

Here we summarize the main contributions of this work.

- We propose new path counting statistics motivated by the cycle consistency structure of the multi-object matching problem formulation. These statistics are designed to yield separated values for good and bad keypoint matches, so that a hard thresholding method can be easily applied. Most notably, we propose a novel way to incorporate cross-keypoint paths that yields better separation between inliers and outlier values. This is the first methodology developed to be robust at the level of keypoint matches rather than at the level of partial permutations.
- We demonstrate how to efficiently compute the statistics for massive datasets in a completely decentralized way. The method involves sparse matrix multiplications and can be efficiently parallelized. The time and space complexity of our method is significantly lower than other methods given sparse initial matches. We further propose a novel iterative reweighting procedure to refine our path-counting statistics.
- We propose a novel synthetic model of SfM data that more realistically mirrors real scenarios while allowing control of parameters. Our method is competitive even though it does not require the number of keypoints m as input, unlike other methods.
- Our method achieves state-of-the-art performance on various real datasets in both accuracy and speed. The method improves the estimation of camera location and rotation when applied to the city-scale Photo Tourism database of [26, 30].

1.3. Structure of the Rest of the Paper

Section 2 provides a mathematical setup and, in particular, reviews the PPS problem and a broader setting that we aim to address, where we do not solve for the absolute partial permutations. Section 3 describes new statistics for removing bad keypoint matches and a practical algorithm that applies them. We motivate these statistics by explaining the geometric structure and, more specifically, cycle-consistency of the underlying graph. Section 4 gives experiments on synthetic and real data that demonstrate the utility and efficiency of these statistics. Finally, §5 concludes this work, while discussing open directions.

2. Problem Setup

We assume n images I_1, ..., I_n of a 3D scene, m scene points, and that an algorithm has identified $m_i \leq m$ keypoints in each image I_i, $i \in [n]$ that aim to describe a subset of the m scene points. Let $\boldsymbol{X}_i^* \in \mathbb{R}^{m_i \times m}$ describe the ground-truth matches between scene points and keypoints in image I_i. More precisely, its kl-th entry is 1 if scene point l corresponds to image keypoint k in image I_i, and 0 otherwise. Note that $\boldsymbol{X}_i^*$ is a partial permutation matrix, that is, it is binary with at most one nonzero element at each row and column. Here in the rest of our notation we use the $*$ superscript to designate ground-truth information, which is unknown to the user.

Given images I_i and I_j, we denote the partial permutation that matches keypoints between these images by $\boldsymbol{X}_{ij}$. Its klth element is 1 if keypoint k in image i corresponds to keypoint l in image j and 0 otherwise. We think of it as an estimate of the ground-truth partial information $\boldsymbol{X}_{ij}^* = \boldsymbol{X}_i^* \boldsymbol{X}_j^{*\top}$. We denote by N the total number of keypoints across all images and form a block matrix for the total keypoint matching $\boldsymbol{X} = (\boldsymbol{X}_{ij})_{i,j=1,\dots n} \in \{0,1\}^{N \times N}$. Similarly, we denote $\boldsymbol{X}^* = (\boldsymbol{X}_{ij}^*)_{i,j=1,\dots n} \in \{0,1\}^{N \times N}$. In the input to our problem, two keypoints in the same image are never connected, and thus we set the block diagonal regions of $\boldsymbol{X}^*$ and $\boldsymbol{X}$ to be 0.

The PPS formulation for multi-object matching asks to estimate the ground-truth absolute partial permutations $\{\boldsymbol{X}_i^*\}_{i=1}^n$ given $\boldsymbol{X}$, or equivalently, finding the ground-truth relative partial permutation $\boldsymbol{X}^*$ from $\boldsymbol{X}$. However, for large and sparse $\boldsymbol{X}$, the corresponding ground truth $\boldsymbol{X}^*$ can be much denser than

$\boldsymbol{X}$, which makes standard PPS algorithms slow and memory-demanding. Moreover, for SfM, the overly dense matches may largely increase the computational burden of robust algorithms for fundamental matrix estimation. Thus, it is sufficient and in fact natural to try to use the good matches that already exist in the given sparse matches $\boldsymbol{X}$. For this reason, our method aims to solve for the intersection between $\boldsymbol{X}^*$ and $\boldsymbol{X}$. Namely, we seek the sparse matrix of good relative matches $\boldsymbol{X}_g := \boldsymbol{X}^* \odot \boldsymbol{X}$, where $\odot$ is the elementwise (Hadamard) product.

We form a graph $G = ([N], E)$ whose nodes in $[N]$ ($[N] = \{1,...,N\}$) index the set of all keypoints in all images and whose edges represent matches between these keypoints. Such a graph was used before in [8, 24, 28]. The matrix $\boldsymbol{X}$ is the adjacency matrix for this graph, and we note that this graph is different from the common graph for PPS [4, 9, 31], whose nodes correspond to images.

Our corruption model assumes within G good (correct) and bad (incorrect) keypoint matches. We thus partition the set of edges into two parts $E = E_g \cup E_b$: E_g denotes the good edges and E_b denotes the bad edges. The corresponding good and bad graphs are $G_g([N], E_g)$ and $G_b([N], E_b)$, respectively. Let $\boldsymbol{X}_g$ and $\boldsymbol{X}_b$ be the adjacency matrices of $G_g([N], E_g)$ and $G_b([N], E_b)$ with blocks $\{\boldsymbol{X}_{ij,g}\}_{i,j=1}^n$ and $\{\boldsymbol{X}_{ij,b}\}_{i,j=1}^n$, respectively, so that $\boldsymbol{X} = \boldsymbol{X}_g + \boldsymbol{X}_b$ and similarly $\boldsymbol{X}_{ij} = \boldsymbol{X}_{ij,g} + \boldsymbol{X}_{ij,b}$ for $i, j \in [n]$. As above, the block diagonal regions of $\boldsymbol{X}_g$ and $\boldsymbol{X}_b$ are always 0.

One can view this model as elementwise-corruption of $\boldsymbol{X}^*$, where each $ij \in E$ $\boldsymbol{X}_{ij}$ is potentially corrupted, instead of the inlier-outlier corruption model [4] that assumes corruption of $\boldsymbol{X}^*$ at the block level. We are not aware of any previous PPS work that focus on this elementwise-corruption model. In view of this model and the above discussion, our problem is the estimation of $\boldsymbol{X}_g$ given the measurement matrix $\boldsymbol{X}$. In simple words, we seek to detect good keypoint matches within $\boldsymbol{X}$. Examples that further demonstrate this setting appear in the supplementary material.

In the following, we will develop novel statistics to filter bad edges in G. To formalize our ideas, we use a few different graphs, we summarize them below in Table 1, even though the last two are defined later.

Graph	Definition	Adj.
$G_g([N],E_g)$	Good keypoint matches	$\boldsymbol{X}_g$
$G_b([N],E_b)$	Bad keypoint matches	$\boldsymbol{X}_b$
$G([N],E)$	Observed keypoint matches	$\boldsymbol{X}$
$G_D([N],E_D)$	Within image matches (see (3))	$\boldsymbol{D}$
$\hat{G}^*([N],\hat{E}^*)$	Minimal cycle-consistent graph containing G_g	$\hat{\boldsymbol{X}}^*$

Table 1. Graphs used throughout the paper. Adj. is an abbreviation for adjacency matrix. All graphs are defined on the same set of nodes $[N]$, which indexes the set of keypoints across all images.

3. Novel Statistics for Removing Bad Matches

Our method aims to remove bad keypoint matches through novel statistics not yet exploited in the literature, and we accomplish this by examining the structure of the graph G described in §2. In §3.1, we describe the notion of cycle consistency and how it fits with our graph G. Then, in Sections 3.2 and 3.3, we formulate the novel statistics, $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, that take advantage of two different consistency constraints. In §3.4, we show how we combine these statistics. The supplementary material illustrates the usefulness of the proposed statistics and their combination for a simple motivating example. Finally, §3.5 discusses the computation of these statistics in practice.

3.1. Cycle Consistency

We utilize the notion of *cycle consistency* to filter out bad keypoint matches. We cannot use the common notion of cycle consistency in partial permutation synchronization [4], since we consider a graph whose nodes represent keypoints instead of images. As we discuss below, we find it more convenient to define a *cycle-consistent graph* instead of a *consistent cycle*. This notion of cycle consistency was explored before in works such as [8, 24].

We say that a graph $G'(V', E')$, where $V' \subset \mathbb{N}$, is cycle-consistent if, whenever $i,j,k \in V'$ and $ik,kj \in E'$, then $ij \in E'$. This definition uses a 3-cycle $\{ij,jk,ki\}$, but one can note that the same property holds for higher-order cycles. We verify this claim for 4-cycles, where the extension to higher-order cycles easily follows by induction. Given $i,j,k,l \in V'$ and assuming that ij, jk, $kl \in E'$, then the original definition implies that $ik \in E'$, and since $kl \in E'$, then also $li \in E'$, that is, the 4-cycle $\{ij,jk,kl,li\}$ is in E'. We easily note that this definition and its extension to higher-order cycles immediately imply that cycle-consistent graphs are dense in the following sense:

Proposition 1. The connected components of any cycle-consistent graph are complete subgraphs.

For any graph of good keypoint matches, $G_g([N],E_g)$, one may extend the set E_g and complete its missing edges in each connected component. This naturally results in the smallest cycle-consistent graph $\hat{G}^*([N],\hat{E}^*)$ that contains G_g. We denote its adjacency matrix by $\hat{\boldsymbol{X}}^*$ (in §3.2, we clarify when $\hat{\boldsymbol{X}}^* = \boldsymbol{X}^*$). This graph contains the complete information for solving the PPS problem. However, to solve our problem it is sufficient for us to try to only recover its subgraph $G_g([N],E_g)$.

3.2. $\boldsymbol{S}_1$ and Within-Cluster Consistency

Our first statistic $\boldsymbol{S}_1$ for distinguishing between inlier and outlier matches uses "within-cluster consistency". We first define this statistic and then motivate it, while explaining the notion of within-cluster consistency.

Fix a small integer $q \geq 2$, where we later use the default value of 4 in our algorithm. The within-cluster statistic is then defined as

$$\boldsymbol{S}_1 = \boldsymbol{X}^q. \tag{1}$$

For $ij \in E$, $\boldsymbol{S}_1(i,j)$ counts the number of paths of length q connecting nodes $i, j \in [N]$ in $G([N],E)$. As we explain below, we generally expect that

$$\boldsymbol{S}_1(i,j) \geq \boldsymbol{S}_1(k,l)\ \forall ij \in E_g,\ kl \in E_b, \text{ when } \boldsymbol{X} \approx \boldsymbol{X}_g. \tag{2}$$

Indeed, good edges are contained in dense subgraphs in the ideal case while bad edges must straddle two dense subgraphs that may not have many connections between them.

The following fact illuminates the above idea:

Fact 3.1. The good subgraph $G_g=([N],E_g)$ and its minimal cycle-consistent extension $\hat{G}^*([N],\hat{E}^*)$ have m' connected components, where $m'\geq m$. Moreover, the rank of $\hat{\boldsymbol{X}}^*$ is m'.

This fact follows from the observation that keypoints of different scene points are not connected in G_g. This fact further implies that a good keypoint match, $ij\in E_g$, should be within a cluster (a dense subgraph). Therefore there should be many short paths connecting ij. On the contrary, a bad match that corresponds to a bad edge ij should be between two clusters, and therefore there should be fewer short paths connecting ij.

Ideally, $m=m'$, in which case $\hat{\boldsymbol{X}}^*=\boldsymbol{X}^*$, but in practice m' might be larger as some disconnected subclusters may occur due to the sparsity of the observed matches. Fact 3.1 suggests a stochastic block model for the underlying structure of $\boldsymbol{X}$ that can be revealed by spectral methods [19, 4]. These methods often assume a relatively dense G_g so that $m'=m$. Then, by finding a rank m approximation of $\boldsymbol{X}$, one can estimate $\hat{\boldsymbol{X}}^*$. However, m is unknown in practice and the observed $\boldsymbol{X}$ is not exactly low rank due to corruption and sparsity in the observation. Moreover, in large-scale datasets, complete recovery of $\hat{\boldsymbol{X}}^*$ is unnecessary and requires large amounts of memory and computational time.

3.3. S_2 and Cross-Cluster Consistency

We define our second statistic $\boldsymbol{S}_2$ and then motivate it in view of what we call cross-cluster consistency. We arbitrarily fix two nonnegative integers r,s such that $r+s=q$ (where q was fixed in §3.2, and our default values are $r=s=2$). Let $\boldsymbol{D}$ be a block diagonal matrix with the same block sizes as $\boldsymbol{X}$, whose diagonal blocks are

$$\boldsymbol{D}_{ii}=\mathbf{1}_{m_i}\mathbf{1}_{m_i}^T-\boldsymbol{I}_{m_i},\text{ for } i\in[n] \tag{3}$$

where $\mathbf{1}_{m_i}$ denotes a column vector of ones in $\mathbb{R}^{m_i}$ and $\boldsymbol{I}_{m_i}$ denotes the $m_i\times m_i$ identity matrix. Let G_D denote the graph whose adjacency matrix is $\boldsymbol{D}$. This graph connects all pairs of distinct keypoints that belong to the same image, which is in contrast to the graph G connects points across different images. We define our second statistic as

$$\boldsymbol{S}_2=\boldsymbol{X}^r\boldsymbol{D}\boldsymbol{X}^s. \tag{4}$$

Note that for $ij\in E$, $\boldsymbol{S}_2(i,j)$ is the number of paths of length $q+1$ connecting nodes i and j composed of an r-length path in $G([N],E)$, then an edge in E_D that connects nodes in the same image, and then an s-length path in $G([N],E)$. Due to this interpretation and the fact stated below we expect that

$$\boldsymbol{S}_2(i,j)=0\leq\boldsymbol{S}_2(k,l)\ \forall ij\in E_g,\ kl\in E_b,\text{ when }\boldsymbol{X}\approx\boldsymbol{X}_g. \tag{5}$$

Fact 3.2. The good subgraph G_g and its dense version $\hat{G}^*$ do not contain any path that connects two keypoints in the same image.

This fact is obvious as such a path would require either a bad keypoint match or a within image match. One thus expects that for $ij\in E_g$ $\boldsymbol{S}_2(i,j)=0$ as otherwise there is a path with an edge in G_D that connects between two disconnected subgraphs of G_g. To the best of our knowledge, this fact has not yet been fully utilized and this turns out to be key for our method.

To quantify the above idea more precisely, we give the following proposition, whose proof is in the supplemental material.

Proposition 2. If $G_{\text{sub}}([N],E_{\text{sub}})$ is a subgraph of $\hat{G}^*([N],\hat{E}^*)$ with adjacency matrix $\boldsymbol{Y}$ and $\boldsymbol{S}_2^{\text{sub}}:=\boldsymbol{Y}^r\boldsymbol{D}\boldsymbol{Y}^s$, then

$$\boldsymbol{S}_2^{\text{sub}}\odot\hat{\boldsymbol{X}}^*=\mathbf{0}. \tag{6}$$

Proposition 2 implies that $\boldsymbol{S}_2^{\text{sub}}(i,j)>0$ is a sufficient condition for $\hat{\boldsymbol{X}}^*(i,j)=0$, that is, for ij being a bad edge. The next proposition shows that under some assumptions, it is also a necessary condition. Again, its proof can be found in the supplemental material.

Proposition 3. Let G_{sub}, $\boldsymbol{Y}$ and $\boldsymbol{S}_2^{\text{sub}}$ be defined as in Proposition 2. Assume that G_{sub} contains the same number of connected components as $\hat{G}^*$ and that, for any two components of G_{sub}, there exists an image that contains at least one node in each component. Then for any (i,j) in the off-diagonal blocks of $\hat{\boldsymbol{X}}^*$, $\hat{\boldsymbol{X}}^*(i,j)=1$ if and only if $\boldsymbol{S}_2^{\text{sub}}(i,j)=0$.

The additional assumption of Proposition 3 holds when each pair of 3D points are contained in at least one image, in which case the $\boldsymbol{S}_2$ should be very helpful. In cases where the assumption is not satisfied, in particular when two 3D points at opposing ends of a 3D structure cannot be viewed by a single camera, then one may use the within cluster information of $\boldsymbol{S}_1$. This motivates the combination of the two statistics seen in the next section.

3.4. Filtering by Combining S_1 and S_2

We construct our combined statistic as

$$\boldsymbol{S}=(\boldsymbol{S}_1\oslash(\boldsymbol{S}_1+\boldsymbol{S}_2))\odot\boldsymbol{X}, \tag{7}$$

where $\oslash$ denotes elementwise division. That is, for any $ij\in E$, the statistic is the ratio between $\boldsymbol{S}_1$ and the sum of the two statistics, whereas for $ij\notin E$, it assigns zero values.

We generally use $r=s$ due to symmetry, so that paths from i to j and j to i are similarly treated. We choose $q=r+s$ so that $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ have comparable scales, as the number of steps within G is the same. Also, our combined statistic is nicely scaled between 0 and 1. As we show later in (8) and (10), S_1 and S_2 have similar forms, and consequently the sum of $\boldsymbol{S}_1(i,j)$ and $\boldsymbol{S}_2(i,j)$ can be efficiently vectorized. In practice, we recommend $r=2$ (or equivalently $q=4$), and parameter-tuning is not needed. This choice corresponds to walks of length 4, which covers simple paths of length 2 and 4 (thus it also covers the paths of $r=1$). Choosing higher r (longer paths) may increase the computational complexity as $\boldsymbol{X}^r$ can be dense for large r. Furthermore, in the next paragraph we introduce an alternating improvement strategy that allows "message passing" among distant edges, thus longer paths are not needed. Similar strategy is validated in [13] for a different problem.

In view of (2) and (5), we expect that $\boldsymbol{S}(i,j)$ is small for bad edges. In particular, $\boldsymbol{S}(i,j)\in[0,1]$ can in some sense be interpreted as a "probability" that $ij\in E_g$. By replacing $\boldsymbol{X}$ with $\boldsymbol{S}$ in the formulas of $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, our original path counting procedure becomes a weighted path counting, where the weights focus on the clean paths. This observation motivates an iterative procedure, where the path weights and $\boldsymbol{S}$ alternatingly improve each other. In this procedure, the initial input is $\boldsymbol{X}$ and then the input is $\boldsymbol{S}$ obtained at the previous iteration. At the last iteration (or without any iteration) one can then filter the $\boldsymbol{S}$-scores above a certain threshold and identify the edges whose final scores are nonzero as good ones. One can also threshold at each iteration.

For completeness, Algorithm 1 describes this iterative procedure, which we refer to as Filtering by Cluster Consistency (FCC). It uses the notation $\mathbf{1}(\boldsymbol{S}>\tau_t)$ for an $N\times N$ binary matrix whose elements are 1 whenever $\boldsymbol{S}_{ij}>\tau_t$.

Algorithm 1: Filtering by Cluster Consistency (FCC)

Input: $\boldsymbol{X}$ matrix of keypoint matches, T: number of iterations, $\{\tau_t\}_{t=1}^{T}$: threshold in each iteration, τ: threshold for computing the output, q,r,s: statistic powers

Output: $\boldsymbol{Y}$ filtered keypoint matches

$\boldsymbol{Y}\leftarrow\boldsymbol{X}$
for $i=1,\ldots,T$ **do**
  $\boldsymbol{S}_1=\boldsymbol{Y}^q$, $\boldsymbol{S}_2=\boldsymbol{Y}^r\boldsymbol{D}\boldsymbol{Y}^s$
  $\boldsymbol{S}=\boldsymbol{S}_1\oslash(\boldsymbol{S}_1+\boldsymbol{S}_2)\odot\boldsymbol{X}$
  $\boldsymbol{S}=\mathbf{1}(\boldsymbol{S}>\tau_t)$ (optional)
  $\boldsymbol{Y}\leftarrow\boldsymbol{S}$
$\boldsymbol{Y}\leftarrow\mathbf{1}(\boldsymbol{Y}>\tau)$

In practice we find that FCC with soft reweighting (no iterative thresholding) works best in general. In this case, the only parameters left are the number of iterations and the final threshold τ. Allowing different τ's makes FCC a very flexible algorithm, which we explain later in §4.2. We recommend 10 iterations for soft-reweighting. However, there are two cases where iterative-hard thresholding (the optional step) can be useful. First, for highly-corrupted full-permutation synchronization datasets, hard-thresholding can slightly improve the accuracy, as we explain in §4.3. Second, for large datasets, where we want to have minimal passes through the whole data, hard-thresholding the bad edges may accelerate the convergence and fewer iterations are needed (see §4.4). For the midsize datasets we found that at least four iterations with $\tau_t=0.05t$ are sufficient, and for large-size datasets, we have found that only two iterations with $\tau_t=0.1t$ are sufficient.

3.5. On the Complexity of FCC

To calculate our statistics, we note that all matrices in the products are sparse. However, intermediate matrices, such as $\boldsymbol{X}^2\boldsymbol{D}$, may be dense. To solve this issue, we notice that we only need to compute $\boldsymbol{S}$ at elements (i,j) such that $\boldsymbol{X}(i,j)>0$. For this purpose, we use the following formula, which is proved in the supplementary material.

Lemma 1. For any $ij\in E$, and $q=r+s$,

$$\boldsymbol{S}_1(i,j)=\sum_{l\in[n]}\sum_{\substack{k_1=k_2\\ k_1,k_2\in I_l}}\boldsymbol{X}^r(i,k_1)\boldsymbol{X}^s(k_2,j),$$

$$\boldsymbol{S}_2(i,j)=\sum_{l\in[n]}\sum_{\substack{k_1\neq k_2\\ k_1,k_2\in I_l}}\boldsymbol{X}^r(i,k_1)\boldsymbol{X}^s(k_2,j). \tag{8}$$

Note that $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ respectively correspond to the complementary cases $k_1=k_2$ and $k_1\neq k_2$, where k_1 and k_2 are indices of keypoints within the same image. This strong relationship leads to efficient computation of $\boldsymbol{S}$. First, we notice that

$$\boldsymbol{S}_1(i,j)=\langle\boldsymbol{X}^r(:,i),\boldsymbol{X}^s(:,j)\rangle, \tag{9}$$

i.e., $\boldsymbol{S}_1(i,j)$ is the dot product of the ith and jth columns of the symmetric matrices $\boldsymbol{X}^r$ and $\boldsymbol{X}^s$. On the other hand,

$$\boldsymbol{S}_1(i,j)+\boldsymbol{S}_2(i,j)=\sum_{l\in[n]}\left[\left(\sum_{k\in I_l}\boldsymbol{X}^r(i,k)\right)\left(\sum_{k\in I_l}\boldsymbol{X}^s(k,j)\right)\right]. \tag{10}$$

By stacking the sparse elements corresponding to nonzero $\boldsymbol{X}$ values into a vector, (9) and (10) can be efficiently parallelized at the cost of having computed $\boldsymbol{X}^r$ and $\boldsymbol{X}^s$ and sufficiently large memory. To compute (10), we have an additional for-loop over $[n]$, but this is still efficient because n is relatively small in our examples.

Our method becomes more efficient when r and s are sufficiently small so that the powers $\boldsymbol{X}^r$ and $\boldsymbol{X}^s$ are sparse. In particular, if $\mathsf{nnz}(\boldsymbol{X})$ denotes the number of nonzero entries in $\boldsymbol{X}$ and n_q denotes the average non-zero entries of $\boldsymbol{X}^q$ per column, then the time to compute the desired entries of $\boldsymbol{S}_1\odot\boldsymbol{X}$ and $(\boldsymbol{S}_1+\boldsymbol{S}_2)\odot\boldsymbol{X}$ are both $O(\mathsf{nnz}(\boldsymbol{X})(n_r+n_s))$. To precompute $\boldsymbol{X}^r$ and $\boldsymbol{X}^s$, we need $O(\mathsf{nnz}(\boldsymbol{X})\cdot\max(\{n_l:l\leq\max(r,s)-1\}))$. The overall space complexity is $O(N\max(\{n_l:1\leq l\leq\max(r,s)-1\})$. Notice that we have the elementary bound $\mathsf{nnz}(\boldsymbol{X})\leq nN$, and consequently, the time and space complexity of Algorithm 1 are $\leq Nn\max(n_r,n_s)$ and $N(n_r+n_s)$, respectively, and these are worst-case bounds. In the supplementary material, we prove the bound $n_2<n^2$ (for $r=s=2$) for an Erdös-Rényi model with $p=n/N$.

While powers of a matrix may not be sparse in general, we observe that $\boldsymbol{X}^r$ is sparse for r sufficiently small. In particular, we generally use $r=s=2$ in our experiments, and in these cases $\boldsymbol{X}^2$ is observed to be very sparse.

4. Numerical Experiments

We conduct experiments on synthetic and real datasets to verify the effectiveness of the proposed FCC algorithm. In §4.1, we demonstrate on synthetic data the improvement of the classification accuracy by our iterative procedure. Then, in Sections 4.2, 4.3 and 4.4, we test the performance of our method on the EPFL[27]/Middlebury[22], Willow [5] and Photo Tourism [26] datasets. We report the specifications of machines on which we ran the experiments in §A.3 of the supplementary material.

4.1. Convergence of the Iterative Procedure

We generate a novel synthetic dataset that models keypoint matches with 100 3D scene points uniformly distributed on the unit sphere, and 100 Gaussian-distributed cameras. Synthetic keypoints are generated by projecting the 3D points onto the image plane. Two keypoints are connected if and only if they correspond to the same 3D point. To generate the corrupted keypoint matches, with probability 0.5 we independently replace an existing match with a false match. For simplicity, run the default FCC with $q=4$ and $r=s=2$ without thresholding. Figure 1 demonstrates the histogram of the FCC statistics after 1 and 5 iterations. We can see that even though a large fraction of matches are missing and corrupted, the FCC statistic at the first iteration (the values in $\boldsymbol{S}$) already achieves good separation of good and bad matches and there is only a small overlapping area between the two histograms.

Dataset			Input		MatchEig				Spectral				MatchALS				FCC (ours) $\tau=0.5$			FCC (ours) $\tau=0.9$			FCC (ours) $\tau=0.99$			
	n	$\hat{m}$	JD	PR	JD	PR	#M	RT	JD	PR	#M	RT	JD	PR	#M	RT	JD	PR	#M	JD	PR	#M	JD	PR	#M	RT
Dino Ring	48	340	25.9	74.1	44.7	93.6	46	21	32.5	84.4	68	38	26.8	85.0	73	12010	**23.2**	78.6	92	35.8	90.1	57	56.2	**94.0**	35	**3**
Temple Ring	47	396	27.4	72.6	51.8	90.1	41	37	36.1	81.9	66	62	30.3	82.2	73	16137	**25.9**	75.5	94	35.2	88.2	58	49.5	**92.4**	41	**2**
Herz-Jesu-P25	25	517	10.4	89.6	27.0	**94.5**	72	60	21.8	92.3	81	105	18.5	93.3	83	9199	**9.7**	90.7	98	18.1	93.6	83	35.1	94.3	64	**1**
Herz-Jesu-P8	8	386	5.7	94.3	7.1	95.3	96	2	18.6	95.0	84	5	25.1	**95.9**	76	155	**5.4**	94.6	99	12.8	95.6	90	17.4	95.8	84	**<1**
Castle-P30	30	445	28.2	71.8	41.2	85.1	55	60	32.7	80.5	72	98	29.3	80.4	76	13583	**25.5**	76.3	91	35.8	87.3	58	52.0	**89.8**	41	**2**
Castle-P19	19	314	29.9	70.1	43.2	80.4	58	18	32.1	77.8	76	18	34.2	77.0	74	1263	**27.0**	74.2	92	34.4	86.5	59	45.5	**88.2**	47	**<1**
Entry-P10	10	432	24.6	75.4	25.4	81.1	84	19	27.7	81.4	80	22	35.2	77.3	77	322	**24.1**	77.9	94	37.0	88.0	59	45.8	**88.6**	50	**<1**
Fountain-P11	11	374	5.8	94.2	12.3	95.8	90	14	11.1	95.5	92	11	20.2	95.7	82	333	**5.6**	95.0	99	15.0	95.9	87	21.6	**96.4**	79	**<1**

Table 2. Performance on the Middlebury and EPFL datasets. n is the number of cameras; $\hat{m}$, the approximated m, is twice the averaged m_i over $i\in[n]$; JD and PR respectively refer to the Jaccard distance (the lower the better) and the precision rate (the higher the better) in (11) in percentage; #M is the ratio in percentage between the number of refined matches and the number of initial matches; RT is runtime in seconds.

After only 5 iterations, the FCC statistic obtains a clean separation of good and bad matches that nicely concentrate around 1 and 0, respectively. Therefore, thresholding at 0.5 (or in a large interval around it) gives exact classification of good and bad matches. We refer the readers to §A.1 in the supplementary material for details of the synthetic model, and the comparison of speed and accuracy among different algorithms under different model parameters.

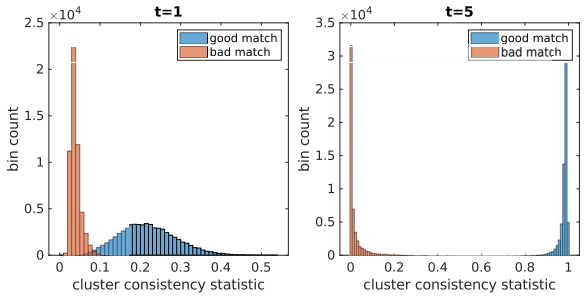

Figure 1. The histograms of the FCC statistics without thresholding for good and bad matches after 1 (left) and 5 (right) iterations.

4.2. Experiments on EPFL and Middlebury

We follow the experimental setup of [16] and compare [19, 16, 31] with our method. Each dataset consists of 8 to 48 images. The Dino Ring and Temple Ring belong to the Middlebury database, and are subsets of the Dino (363 images) and Temple (312 images) datasets. The latter datasets are still much smaller than Photo Tourism datasets in §4.4 due to their small universe size (around hundreds) compared to that of Photo Tourism ($> 10^4$), even though they have similar numbers of images. We do not use the whole datasets of Temple and Dino, since MatchALS cannot handle them and Spectral and MatchEig have similar results on Temple Ring and Dino Ring (see [16]), and running on those subsets is much faster.

The initial matches between pairs of images are generated by nearest neighbor and ratio test using SIFT features followed by RANSAC refinement. The implementation of Spectral [19], MatchALS [31] and MatchEig [16] is exactly the same as in [16]. Since the ground truth size of the universe m is unknown, we follow [16] and approximate m by twice the average of m_i over $i\in[n]$. We denote this approximate size of the universe by $\hat{m}$. Since Spectral, MatchEig and MatchALS require the rank estimate of $\boldsymbol{X}$, we follow [16] and use $\hat{m}$ for Spectral and MatchEig, and $2\hat{m}$ for MatchALS. We note that FCC does not require this parameter. We run FCC with soft reweighting (no iterative thresholding) for 10 iterations. We note that one still needs to threshold the statistics matrix $\boldsymbol{S}$ in the final step to obtain the refined matching. We test FCC with $\tau=0.5,0.9,0.99$. The higher the threshold, the more sparse the resulting match is.

Following [16], when evaluating the estimated matches, a match is good if the epipolar constraint approximately holds given the two keypoints and ground truth camera parameters. We report two types of metrics, the precision rate (PR) and Jaccard distance (JD) of classification:

$$\text{PR}=|\hat{E}\cap E_g|/|\hat{E}|,\quad \text{JD}=1-|\hat{E}\cap E_g|/|\hat{E}\cup E_g|, \qquad (11)$$

where $\hat{E}$ is the estimate of E_g. We note that Jaccard distance is a more balanced metric that considers both precision and recall (note that the Jaccard distance is a decreasing function of the F-score).

We remark that different metrics may fit better different tasks. The PR metric may be more useful for SfM tasks with initial dense matches, since as long as the refined match is good then one can reliably compute fundamental matrices (so one does not care how many good matches were thrown out). However, the combination of both precision and recall may fit better with other tasks, such as permutation synchronization, where all images share the same set of keypoints. In this case, a more natural metric is the JD. In addition to PR and JD, we report the percentage of initial matches that remains after refinement by different algorithms (#M in Table 2), which partially reflects recall. We also report runtimes of different algorithms in seconds (RT in Table 2).

Table 2 indicates superior performance of FCC in comparison to other methods in terms of both accuracy and speed. We first note that FCC is in general 10x - 100x faster than the current fastest approaches MatchEig and Spectral, and is about 5000x - 10000x faster than MatchALS, on the datasets of EPFL and Middlebury. Moreover, FCC with $\tau = 0.5$ achieves the best Jaccard distance, namely the classification error. Choosing higher τ removes more matches, which corresponds to higher precision and lower recall. When choosing $\tau = 0.99$, FCC yields better precision than other methods and still maintains around 50% of

Datasets	n	N	Input	Spectral [19]	PPM [11]	MLift [4]	MALS [31]	IRGCL [25]	FCC	FCC+PPM
Car	40	400	0.52	0.36	0.26	0.26	0.28	0.25	0.31	**0.24**
Duck	50	500	0.57	0.34	0.34	0.33	0.34	**0.30**	0.40	0.35
Face	108	1080	0.14	**0.041**	0.046	0.054	0.055	0.048	0.057	0.046
Motorbike	40	400	0.7	0.65	0.61	**0.57**	**0.57**	0.63	0.64	**0.57**
Winebottle	66	660	0.48	0.29	0.26	0.25	0.25	0.24	0.28	**0.23**

Table 3. Matching performance comparison using the Willow database. Note that $m=10$ and $N=10n$.

matches on most datasets. Thus, depending on the tasks, one can choose either $\tau = 0.5$ or $\tau = 0.99$ (or some other values) to encourage better performance on either Jaccard distance or precision (or balance between the two). For this reason, we find that FCC is a very flexible algorithm that can be suitable for different tasks, and for both choices FCC achieves the best performance than other methods.

We also find that MatchEig generally performs better than Spectral. The reason is that MatchEig is less sensitive to the parameter $\hat{m}$, as explained in [16]. MatchALS is the slowest algorithm, and its error and precision are worse than MatchEig and similar to Spectral. We note that MatchALS only achieves better PR than FCC on Herz-Jesu-P8, where it maintains fewer matches (76%) than FCC (84%).

4.3. Experiments on the Willow Database

We test FCC on the Willow database [5] for multi-object matching. It includes 5 small datasets, where each dataset consists of dozens of images taken from similar viewing directions and each image contains 10 keypoints of the same 10 3D scene points. The ground truth pairwise matches are all 10×10 full-permutation matrices (rows and columns sum to 1), so common permutation synchronization algorithms can be applied. We use the initial matches provided by [29] (these matches were obtained by rounding the similarity matrix of CNN features of the keypoints, however, we do not compare with [29] since it uses additional geometric information from keypoint coordinates). We run FCC with $T = 10$, $\tau_t = 0.05t$ and $\tau = 0.5$. We compared with the following methods for permutation synchronization: Spectral [19], PPM [11], IRGCL [25], MatchLift [4] and MatchALS [31]. Since they have the advantage of using the permutation synchronization model and they use the parameter m, we also tested FCC as an initializer to one of these methods, PPM. We preferred PPM since it seems more sensitive to initialization. We refer to the combined method, which first removes a small fraction of matches with low values of the FCC statistic and then applies PPM, as FCC+PPM. For the combined algorithm, we run FCC with $T = 4$ only, $\tau_t = 0.05t$ and $\tau = 0.1$. By choosing such a low threshold, we only remove the extremely suspicious matches, while keeping the majority of matches. The reason for doing this is that the dataset is extremely noisy, and a large threshold would remove some good edges so that pairwise matches may be too sparse and will not give rise to permutations. Thus a large threshold for FCC will degrade any follow-up algorithm for full-permutation synchronization, for which we use PPM. We use the metric of (11) to measure accuracy. Table 3 summarizes the results. We note that Spectral, PPM and IRGCL directly use the special structure of full-permutations and rely on the Hungarian algorithm to project the estimates to full-permutations. In contrast, FCC does not make these assumptions since it is designed for more general PPS. Without these additional information (m and full-permutations), mere FCC is a reasonable algorithm that is roughly comparable to Spectral. We also note that FCC significantly improves PPM and the combined algorithm outperforms on average the rest of the algorithms.

We remark that the thresholding procedure in FCC is helpful in Willow. The reason is that graphs for the Willow datasets are relatively dense and the FCC statistics for most good matches are strictly above 0. Thus, we take a conservative strategy by using a small threshold (0.05) in the first iteration to make sure that only bad matches are removed and then gradually increase the threshold in each iteration.

4.4. Experiments on the Photo Tourism Database

We test FCC on the SfM Photo Tourism database [26] with precomputed pairwise image matches provided by [23]. These matches were obtained by thresholding SIFT feature similarities. We compare the LUD pipeline [17] to the LUD pipeline with FCC pre-processing (denoted as FCC+LUD). All components not involving FCC were implemented identically in both pipelines.

For FCC+LUD, we first prune those matches with the FCC method using the parameters $T=2$, $\tau_t=0.1t$ and $\tau=0.5$. We remark that typically on such SfM data, we find that most of the values of the $\boldsymbol{S}$ statistic are either larger than 0.9 or smaller than 0.1. Therefore the output is not sensitive to the choice of τ, which is away from 0 and 1.

After filtering keypoint matches for all pairs of images using FCC, we computed the essential matrices using the least median of squares procedure. We did not apply RANSAC since it is sensitive to the choice of threshold and sample size, which introduces more randomness in our evaluation. In contrast, the least median of squares is parameter-free and much faster. We also computed the essential matrices of the LUD pipeline in the same way. To account for cases where there were not enough keypoint matches, if there were less than 16 keypoint matches remaining after FCC filtering (as required by the median least squares), we remove the correspondence between the two images. At last, we fed essential matrices into the camera pose solver in the standard LUD pipeline [17]. We remark that the camera location solver in the LUD pipeline automatically examines the parallel rigidity of the viewing graph and extracts the maximal parallel rigid subgraph. This subgraph extraction procedure often removes more cameras if some camera correspondences are removed by FCC beforehand.

Algorithms			LUD						FCC+LUD						
Dataset	n	N	n	$\hat{e}_R$	$\tilde{e}_R$	$\hat{e}_T$	$\tilde{e}_T$	T_{total}	n	$\hat{e}_R$	$\tilde{e}_R$	$\hat{e}_T$	$\tilde{e}_T$	T_{FCC}	T_{total}
Alamo	570	606963	557	20.90	16.10	8.17	5.18	7945.9	538	**19.16**	**15.23**	**7.81**	**4.92**	755.1	8788.5
Ellis Island	230	178324	223	2.16	1.16	22.99	**21.95**	1839.2	218	**1.87**	**1.02**	**22.78**	22.20	79.6	1904.6
Gendarmenmarkt	671	338800	652	**40.14**	9.30	**38.55**	**18.33**	3527.7	625	40.31	**8.48**	38.82	18.58	122.9	3746.1
Madrid Metropolis	330	187790	315	13.35	9.37	12.80	6.87	1579.8	297	**11.51**	**6.87**	**11.69**	**4.99**	44.2	1615.9
Montreal N.D.	445	643938	439	2.55	1.06	1.51	**0.66**	5078.9	416	**1.72**	**0.89**	**1.34**	0.67	550.5	5630.6
Notre Dame	547	1345766	545	3.72	1.44	1.45	0.41	11315.2	534	**3.46**	**1.40**	**1.35**	**0.40**	4103.3	14965.5
NYC Library	313	259302	306	4.05	2.22	7.09	2.81	1495.9	283	**3.43**	**2.07**	**6.48**	**2.43**	47.6	1536.5
Piazza Del Popolo	307	157971	300	6.92	3.98	6.78	2.42	1989.9	243	**1.76**	**0.90**	**2.36**	**1.30**	74.4	2053.9
Piccadilly	2226	1278612	2015	8.11	3.77	5.41	2.98	21903.3	1928	**7.17**	**3.67**	**5.04**	**2.80**	1190.7	26750.7
Roman Forum	995	890945	971	**6.64**	**5.02**	**12.67**	**5.60**	4858.0	906	6.67	5.39	13.29	5.66	153.6	5315.5
Tower of London	440	474171	431	**6.89**	4.29	21.47	6.85	1759.2	408	7.17	**4.12**	**19.14**	**6.52**	35.5	1825.1
Union Square	733	323933	663	10.77	6.93	**14.52**	10.49	1950.6	599	**8.31**	**5.91**	14.72	**10.12**	33.7	2083.8
Vienna Cathedral	789	1361659	758	6.58	3.12	14.52	8.28	10866.0	682	**3.76**	**1.92**	**11.60**	**6.79**	2845.6	13550.2
Yorkminster	412	525592	407	4.25	2.71	6.45	3.68	2267.3	386	**4.04**	**2.66**	**6.00**	**3.44**	90.1	2371.1

Table 4. Performance on the Photo Tourism datasets: n and N are the number of cameras and key points, respectively (n is listed three times as the initial number for the dataset and the remaining numbers after removing cameras in both pipelines); $\hat{e}_R$ $\tilde{e}_R$ 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_{FCC} and T_{total} are the runtime of FCC and the total runtime of the given pipeline (LUD or FCC+LUD), respectively (in seconds).

However, our experiments show that the final numbers of cameras in both our procedure and pure LUD are comparable (with at least 80% of cameras remaining in each dataset).

Table 4 reports the number of cameras, the accuracy (mean and median errors of rotations and translations) and runtime of the standard LUD pipeline and the new FCC+LUD procedure. From the table, we observe that FCC+LUD improves the estimation of camera parameters over LUD on the unfiltered keypoint matches in a significant portion of the datasets. In particular, the only two sets where the LUD pipeline has better accuracy overall are Gendarmenmarkt (where the error is very high anyway) and Roman Forum (where the differences are not too significant). This is mainly due to the highly symmetric buildings contained in the two datasets which results in self-consistent bad keypoint matches. These malicious matches cannot be removed by merely exploiting the cycle-consistency, and 3D geometric information should be used to solve this ambiguity.

One may also compare the total runtime of LUD with that of FCC+LUD and its subcomponent, FCC. For most datasets the sum of the total time of LUD and of FCC is about the same as the total time of FCC+LUD, although the one exception is the Piccadilly dataset. We noticed that in this case it takes much more time to extract the maximal parallel rigid component (it takes 4000 seconds for FCC+LUD, while only 72 seconds for LUD). For pure LUD, the parallel rigid component seems close to the original graph; however, this is not the case for FCC+LUD.

Additional figures appear in the supplemental material. One set of figures compares the estimation errors of FCC+LUD and LUD on their common set of cameras. They demonstrate that FCC+LUD generally performs better than LUD on this set of cameras, so the advantage of FCC+LUD was not just obtained by removing bad cameras. The other set of figures compares the performance of LUD on the cameras of FCC+LUD and the rest of the cameras. The errors on the first set of cameras are generally smaller than on the second set. Thus, FCC is helpful in removing bad cameras. We finally remark that all other standard PPS algorithms are not scalable to the Photo Tourism datasets, and thus we only apply FCC here.

5. Conclusion

In this work, we develop novel robust statistics for multi-object matching. These statistics are based on cycle consistency constraints on the graph with all image keypoints as nodes and keypoint matches as edges. In particular, we combine within-cluster and cross-cluster constraints into a combined statistic to yield distinguished values between corrupted and uncorrupted matches. The resulting FCC method is efficiently implementable in practice due to only requiring sparse matrix multiplication and parallelization. Experiments in synthetic and real data demonstrate state-of-the-art accuracy and speed for structure from motion tasks. In particular, FCC is the only robust multi-object matching method that can scale to city-scale SfM data.

Due to the performance of our method combined with the intriguing heuristics, one direction for future work is to theoretically explore the properties of our statistics. Further, future work will explore the incorporation of SIFT descriptor similarities in our statistics, instead of using the adjacency matrix of the keypoint matching graph alone, in order to yield even more accurate 3D reconstructions. Finally, the use of the cross-cluster constraints allows us to develop a complementary statistic that has not been thought of before in the literature. In particular, this complementary statistic is based on considering paths containing edges in the disjoint graphs G and G_D. It would be interesting to explore applications of such disjoint graph constraints in other settings.

Acknowledgement

This work was supported by NSF award DMS 1821266.

References

[1] Florian Bernard, Johan Thunberg, Jorge Goncalves, and Christian Theobalt. Synchronisation of partial multi-matchings via non-negative factorisations. *Pattern Recognition*, 92:146–155, Aug 2019. 2

[2] Florian Bernard, Johan Thunberg, Paul Swoboda, and Christian Theobalt. Hippi: Higher-order projected power iterations for scalable multi-matching. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10283–10292. IEEE, 2019. 2

[3] 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. 2

[4] 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. 1, 2, 3, 4, 7, 10

[5] Minsu Cho, Karteek Alahari, and Jean Ponce. Learning graphs to match. In *Proceedings of the IEEE Interational Conference on Computer Vision*, 2013. 5, 7

[6] Michael R Garey, David S Johnson, and Larry Stockmeyer. Some simplified np-complete problems. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63. ACM, 1974. 1

[7] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision (2. ed.)*. Cambridge University Press, 2006. 1

[8] Nan Hu, Qixing Huang, Boris Thibert, and Leonidas J Guibas. Distributable consistent multi-object matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2463–2471, 2018. 1, 2, 3, 10, 11

[9] Qi-Xing Huang and Leonidas J. Guibas. Consistent shape maps via semidefinite programming. *Comput. Graph. Forum*, 32(5):177–186, 2013. 3

[10] Qi-Xing Huang, Guo-Xin Zhang, Lin Gao, Shi-Min Hu, Adrian Butscher, and Leonidas Guibas. An optimization approach for extracting and encoding consistent maps in a shape collection. *ACM Trans. Graph.*, 31(6):167:1–167:11, 2012. 1

[11] Vahan Huroyan. *Mathematical Formulations, Algorithm and Theory for Big Data Problems*. PhD thesis, University of Minnesota, 2018. 2, 7, 10

[12] Spyridon Leonardos, Xiaowei Zhou, and Kostas Daniilidis. A low-rank matrix approximation approach to multiway matching with applications in multi-sensory data association. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8665–8671, 2020. 2

[13] Gilad Lerman and Yunpeng Shi. Robust group synchronization via cycle-edge message passing. *Foundations of Computational Mathematics*, 2021. 1, 4

[14] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690, 2007. 1

[15] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 1

[16] 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. 2, 6, 7

[17] Onur Özyesil and Amit Singer. Robust camera location estimation by convex programming. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 2674–2683, 2015. 7

[18] Onur Özyesil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion. *Acta Numerica*, 26:305–364, 2017. 1

[19] Deepti Pachauri, Risi Kondor, and Vikas Singh. Solving the multi-way matching problem by permutation synchronization. In *Advances in Neural Information Processing Systems 26*, pages 1860–1868. 2013. 1, 2, 4, 6, 7, 10

[20] Amelia Perry, Alexander S. Wein, Afonso S. Bandeira, and Ankur Moitra. Message-passing algorithms for synchronization problems over compact groups. *Communications on Pure and Applied Mathematics*, 2018. 1

[21] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, July 1976. 1

[22] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006),*, pages 519–528. IEEE Computer Society, 2006. 5

[23] 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. 7

[24] Zachary Serlin, Guang Yang, Brandon Sookraj, Calin Belta, and Roberto Tron. Distributed and consistent multi-image feature matching via quickmatch. *The International Journal of Robotics Research*, 39(10-11):1222–1238, 2020. 1, 2, 3, 10

[25] Yunpeng Shi, Shaohan Li, and Gilad Lerman. Robust multi-object matching via iterative reweighting of the graph connection Laplacian. In *Advances in Neural Information Processing Systems*, volume 33, pages 15243–15253, 2020. 2, 7

[26] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*, pages 835–846, New York, NY, USA, 2006. ACM Press. 2, 5, 7

[27] Christoph Strecha, Wolfgang von Hansen, Luc Van Gool, Pascal Fua, and Ulrich Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society, 2008. 5

[28] Roberto Tron, Xiaowei Zhou, Carlos Esteves, and Kostas Daniilidis. Fast multi-image matching via density-based clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 4057–4066, 2017. 1, 3

[29] 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. 2, 7

[30] 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. 1, 2

[31] Xiaowei Zhou, Menglong Zhu, and Kostas Daniilidis. Multi-image matching via fast alternating minimization. In *IEEE International Conference on Computer Vision, ICCV 2015*, 2015. 1, 2, 3, 6, 7, 10