Distributable Consistent Multi-Object Matching

Nan Hu
Stanford University

Qixing Huang
UT Austin

Boris Thibert
UG Alpes

Leonidas Guibas
Stanford University

Abstract

In this paper we propose an optimization-based framework to multiple object matching. The framework takes maps computed between pairs of objects as input, and outputs maps that are consistent among all pairs of objects. The central idea of our approach is to divide the input object collection into overlapping sub-collections and enforce map consistency among each sub-collection. This leads to a distributed formulation, which is scalable to large-scale datasets. We also present an equivalence condition between this decoupled scheme and the original scheme. Experiments on both synthetic and real-world datasets show that our framework is competitive against state-of-the-art multi-object matching techniques.

1. Introduction

Object matching techniques have been widely used in many fields of computer vision, including 2D and 3D image analysis, object recognition, biomedical identification, and object tracking. There is a rich literature on finding meaningful approximate isomorphisms between pair of objects that are represented as graphs [24, 6, 9, 10, 26, 8, 13, 14]. Many tasks, however, require to solve the so-called multi-object matching problem, i.e., finding consistent maps among all pairs of objects within a collection. Examples include non-rigid structure from motion [1, 7] and shared object discovery [3]. In this context, a central task is how to utilize the data collection as a regularizer to improve the maps computed between pairs of objects in isolation [15, 4, 32].

A generic constraint that one can utilize to improve maps among a collection is the so-called *cycle consistency* constraint, namely composition of maps along any two paths sharing the same starting and end objects are identical. A technical challenge of utilizing this constraint is that it is impossible to check all cycles for consistency, due to the fact that the number of paths increase exponentially with the total number of objects. Recent works on joint matching have shown that the cycle consistency constraint can be translated into a much more manageable constraint, i.e., the data matrix that stores pair-wise maps in blocks is positive semidefinite and low-rank [15, 22]. Based on this connection, people have formulated multi-object matching as solving semidefinite programs (or SDP), which are convex relaxations of the corresponding matrix recovery problem. These algorithms achieved near-optimal exact recovery conditions [15, 4]. On the other hand, solving semidefinite programs are computationally expensive. In a recent work, Zhou et al [32] attempt to address the computational issue using alternating minimization for efficient low-rank matrix recovery.

In this paper, we propose a novel framework that utilizes the cycle-consistency constraint in a hierarchical manner for scalable multiple object matching. We show how to apply this framework to extend the methods described in [15, 4, 32]. In particular, instead of jointly imposing the global consistency constraint among all pair-wise maps [15, 4, 32], we split the input object collection into overlapping subsets, and impose consistency within each subset. We then impose consistency between maps across the subsets. Interestingly, we show that by combing these two consistency constraints together, we can guarantee global consistency under mild conditions (See Section 2). Yet computationally, such a decoupled approach yields significant performance gains, when compared with existing approaches.

1.1. Related Work

Early works on multi-object matching (e.g., [31, 16]) extend pairwise matching schemes to the multi-object setting without explicitly considering the map consistency constraint. [30, 20] proposed to detect inconsistent cycles, and formulate multi-object matching as solving combinatorial optimizations, i.e., removing bad maps to break all inconsistent cycles. Recently, people have proposed to formulate non-convex optimization problems by using the cycle consistency constraint as an explicit constraint for either pixel-wise flow computation [32], sparse feature matching [27], sparse shape modeling [5], or structure from motion [21]. These problems are, as a consequence, hard to solve and do not admit exact recovery conditions. Recent works [15, 22] showed that consistent maps could be extracted from the spectrum of a data matrix that encodes pair-wise maps in blocks. Along this line of research, Huang and Guibas [15] proposed an elegant solution by formulating the problem as convex relaxation and discussed the theoretical conditions for exact recovery. The result is further analyzed in [4] under the condition that the underlying rank of the variable matrix is known or can be reliably estimated. Yan et al. [29, 28] also proposed matrix factorization based methods to enforce the cycle-consistency constraint. These methods, however, are not scalable to large-scale datasets, due to the cost of solving semidefinite programs. Zhou et al. [32] enforce the positive semidefinite constraint using explicit low-rank factorizations, leading to improved computational efficiency. In contrast to these methods, our approach opens a new direction to enforcing the cycle-consistency constraint, i.e., by splitting the datasets into overlapping subsets. This leads to further improvements in terms of computational efficiency. Most recently, Leonardos et. al. [17] proposed a distributed consensus-based algorithm as an extension of [22]. Their method, however, cannot handle partial matches.

We organize the reminder of this paper as follows. First, we discuss the problem setup and analyze the conditions in Section 2. Second, we discuss the formulation of our approach in Section 3. In Section 4, we present an alternating direction method of multipliers (ADMM) for solving the induced optimization problem, leading to a parallel algorithm via generalized message passing. Last but not the least, we demonstrate the effectiveness of our approach on both synthetic and real examples in Section 5.

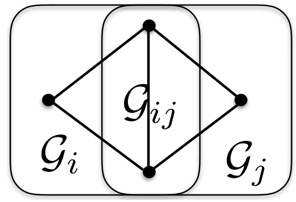
2. Consistency

In this section, we extend the cycle-consistency formulation described in [15] to the distributed setting. The key result is a sufficient condition on which cycle-consistency among sub-collections induces global cycle-consistency.

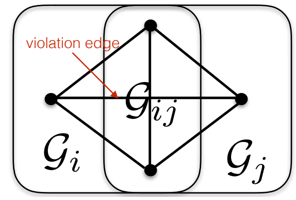
We begin with introducing the notations that are necessary to formally state this sufficient condition. For simplicity, we assume maps between objects are given by permutations. However, the argument can be easily extended to the case where objects are partially similar with each other. Formally speaking, we consider a map graph $\mathcal{G} = (\mathcal{V} = (H_1, \cdots, H_n), \mathcal{E})$. The vertex set $\mathcal{V}$ consists of objects to be matched, and each object H_i is given by m points (e.g., key points extracted from an image). The edge set $\mathcal{E}$ connects a subset of pairs of objects. Each edge $(i,j) \in \mathcal{E}$ is associated with a permutation $\phi_{ij} : H_i \to H_j$. We first define the global consistency of $\phi_{ij}, \forall (i,j) \in \mathcal{E}$:

Definition 1 (Cycle Consistency)**.** *A map graph* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ *is* cycle consistent *if for every node* v_i *and every cycle* $v_i - v_{i_1} - \cdots - v_{i_k} - v_i$*, the composite map along this cycle is the identity map, i.e.*

$$\phi_{ii_1} \circ \cdots \circ \phi_{i_k i} = \text{identity}.$$



(a) *joint normal*



(b) Counter *joint normal*

Figure 1: Examples of subgraphs that are (a) joint normal and (b) not joint normal.

Now we introduce an equivalent formulation of enforcing the cycle-consistency among $\mathcal{G}$ as enforcing the cycle-consistency among subgraphs of $\mathcal{G}$, if these subgraphs satisfy certain conditions. Towards this end, we introduce two conditions among collection of subgraphs of $\mathcal{G}$. The first condition concerns a pair of sub-graphs:

Definition 2 (Joint Normal)**.** *Let* $\mathcal{G}_i = \{\mathcal{V}_i, \mathcal{E}_i\}$, $\mathcal{G}_j = \{\mathcal{V}_j, \mathcal{E}_j\}$ *be the two subgraphs of* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$*. We say* $\mathcal{G}_i$ *and* $\mathcal{G}_j$ *are joint normal if the vertex sets* $\mathcal{V}_i \setminus \mathcal{V}_j$ *and* $\mathcal{V}_j \setminus \mathcal{V}_i$ *are not connected by any edge of* $\mathcal{E}$*:*

$$(s,t) \notin \mathcal{E}, \ \forall H_s \in \mathcal{V}_i \backslash \mathcal{V}_j, H_t \in \mathcal{V}_j \backslash \mathcal{V}_i$$

As illustrated in Figure 1, two subgraphs $\mathcal{G}_i = \{\mathcal{V}_i, \mathcal{E}_i\}$ and $\mathcal{G}_j = \{\mathcal{V}_j, \mathcal{E}_j\}$ are *joint normal* if their common subgraph $\{\mathcal{V}_i \cap \mathcal{V}_j, \mathcal{E}_i \cap \mathcal{E}_j\}$ is either (i) empty, or (ii) connected, and there is no edge between a vertex in one subgraph to a vertex in the other subgraph except those in the common subgraph. In contrast, the two subgraphs illustrated in Figure 1(b) are not joint normal since there exists an edge that connects the non-overlapping sets of these two subgraphs.

The second condition concerns a topological constraint among all the sub-graphs. We state this condition using the notation of simplicial complex as detailed below:

Definition 3 (Cover Complex)**.** *Let* $\{\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i), 1 \leq i \leq K\}$ *be a set of sub-graphs that cover* $\mathcal{G}$*, i.e.* $\cup_{i=1}^{K} \mathcal{V}_i = \mathcal{V}$*. We define the cover complex* $\mathcal{K}$ *of these sub-graphs* $\{\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i), 1 \leq i \leq K\}$ *so that* $\mathcal{K}$ *collects every subset* $\{i_1, \cdots, i_k\} \subset \{1, \cdots, K\}$ *if the intersections of* $\mathcal{V}_{i_1}, \cdots, \mathcal{V}_{i_k}$ *is non-empty, i.e.* $\cap_j \mathcal{V}_{i_j} \neq \varnothing$*.*

We now state the decoupled cycle-consistency theorem that relates the global cycle consistency and the cycle consistency on each subgraph:

Theorem 1 (Decoupled Cycle-Consistency)**.** *Let* $\mathcal{G}$ *be a map graph,* $\mathcal{G}_1, \cdots, \mathcal{G}_K$ *be a cover of* $\mathcal{G}$*, and* $\mathcal{K}$ *be the cover complex. Then* $\mathcal{G}$ *is cycle consistent if*

1. $\mathcal{G}_i$ *is* cycle consistent $\forall i$,
2. $\mathcal{G}_i$ *and* $\mathcal{G}_j$ *are* joint normal $\forall (i,j) \in \mathcal{E}$,
3. $\mathcal{K}$ *is* simply connected *(c.f. [12])*.

Here we say the cover complex $\mathcal{K}$ is *simply connected* if every closed curve can be deformed to a point (or in other words the region enclosed by this curve has no-holes). Please refer to [12] for a more general definition. This theorem states that the cycle consistency property on each subgraph would be propagated to the global consistency, if the cover complex $\mathcal{K}$ satisfies the conditions stated in Theorem 1. The proof to Theorem 1 is left to the supplementary material.

Note that the 3rd condition in Theorem 1 is necessary. Figure 2(a) provides a simple counter example, which satisfies the 1st and 2nd conditions in Theorem 1. The cover complex $\mathcal{K}$, however, is homologous to the Torus T^2, which is not simply connected. It is easy to see that the local-consistency (which is trivial as each sub-graph is given by an edge) does not lead to the global consistency among these three edges.

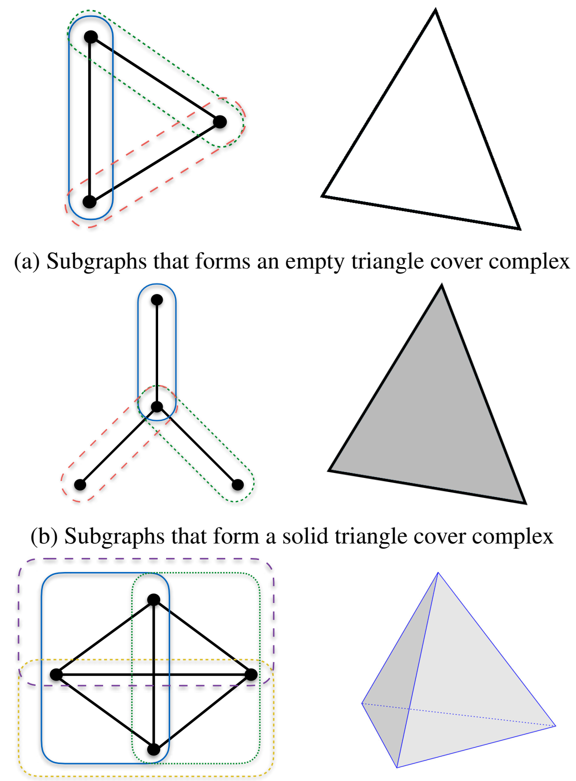
Figure 2(c) provides another example to understand the correctness of Theorem 1. In this case, there are four objects. It is clear that enforcing the cycle-consistency among all four triple sub-graphs induces the cycle-consistency on the original graph. This argument aligns with Theorem 1 as $\mathcal{K}$ is simply connected. We defer detailed explanations to the supplementary material.

Algorithm 1: Greedy Construction of $\mathcal{K}$

Input : Map graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Number of cover nodes K
Output: Cover nodes $\{\mathcal{G}_i\}_1^K$

Compute a graph clustering of $\mathcal{G}$ to K clusters (graph cut or K-means on graph embeddings)
Assign each of the cluster to $\mathcal{G}_i,\ i = 1, \cdots, K$
while *condition not satisfied* **do**
  Assign H_j to $\mathcal{G}_i$ if H_j is neighboring to $\mathcal{G}_i$ in $\mathcal{G}$ or within distance of ϵ to $\mathcal{G}_i$ in the embedding space
  Build $\mathcal{K}$ from $\{\mathcal{G}_i\}_i^K$
  Check if $\mathcal{K}$ is connected
  Compute homology group using [33]
  Check if $\mathcal{H}_1(\mathcal{K})$ is trivial
  if *Both conditions satisfied* **then**
    Break
  end
end

To develop an algorithm based on Theorem 1, we proposed a greedy algorithm to construct $\mathcal{K}$ as in Algorithm 1. Note that a complex $\mathcal{K}$ is *simply connected*, if 1) it is connected; 2) the 1-dimensional homology group $\mathcal{H}_1(\mathcal{K})$ is trivial. Condition 1) could be easily verified by any graph traversal algorithm (BFS/DFS), and condition 2) can be verified computationally as in [33].



(a) Subgraphs that forms an empty triangle cover complex

(b) Subgraphs that form a solid triangle cover complex

(c) Subgraphs that form an empty tetrahedron cover complex

Figure 2: Given local consistency, (a) is not globally consistent, while (b) and (c) are guaranteed to be globally consistent. (Left: the map graph with subgraphs circled out. Right: the corresponding cover complex.)

3. Distributed Optimization

In this section, we introduce the proposed distributed formulation of recovering cycle-consistent maps from noisy pair-wise maps.

3.1. Formulation

Our formulation takes as input the pairwise base maps ϕ_{ij}. We follow the state-of-the-art work on convex relaxation of enforcing cycle-consistent maps [15, 4] to encode ϕ_{ij} into a data matrix $\overline{\mathbf{X}}_{ij}$. Following the common strategy for optimizing point-based maps, we relax ϕ_{ij} to be a partial map and/or soft map, i.e. $\overline{\mathbf{X}}_{ij} \in [0,1]^{m_i \times m_j}$, where m_i denote the number of vertices in H_i.

Let $\overline{\mathbf{X}}_{\mathcal{V}}$ be the matching matrix that encodes pair-wise maps in its blocks, i.e.

$$\overline{\mathbf{X}}_{\mathcal{V}} = \begin{pmatrix} \mathbf{I} & \overline{\mathbf{X}}_{12} & \cdots & \overline{\mathbf{X}}_{1n} \\ \overline{\mathbf{X}}_{21} & \mathbf{I} & \cdots & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ \overline{\mathbf{X}}_{n1} & \cdots & \cdots & \mathbf{I} \end{pmatrix}$$

The goal of our formulation is to find a matrix $\mathbf{X}_{\mathcal{V}}$ that encodes cycle-consistent maps from the noisy input $\overline{\mathbf{X}}_{\mathcal{V}}$. To achieve this goal, one observation is that the desired matching matrix $\mathbf{X}_{\mathcal{V}}$ is low-rank (c.f. [15]). Specififcally, we assume there is an universal object of size m, i.e. there are totally m distinct entities for all the objects H_i's in $\mathcal{V}$. For each object H_i, we have a latent map encoded by $\mathbf{A}_{H_i} \in \{0,1\}^{m_i \times m}$, which maps a vertex from H_i to an entity in the universal object. Let $\mathbf{A}_{\mathcal{V}}$ be a tall matrix that concatenates $\mathbf{A}_{H_i}$, i.e., $\mathbf{A}_{\mathcal{V}} = (\mathbf{A}_{H_1}^T, \cdots, \mathbf{A}_{H_n}^T)^T$. It is easy to see that the map matrix $\mathbf{X}_{\mathcal{V}}$ admits a low-rank factorization as $\mathbf{X}_{\mathcal{V}} = \mathbf{A}_{\mathcal{V}}\mathbf{A}_{\mathcal{V}}^\top$. In [15, 32], the authors use this property to develop robust algorithms for recovering $\mathbf{X}_{\mathcal{V}}$ from noisy input maps.

Without losing generality. we assume $h(\mathbf{X}_{\mathcal{V}})$ is an objective function that measures the quality of a set of cycle-consistent maps encoded by $\mathbf{X}_{\mathcal{V}}$, e.g., it promotes the low-rankness of $\mathbf{X}_{\mathcal{V}}$. The precise expression of $h(\mathbf{X}_{\mathcal{V}})$ will be discussed later. Our distributed formulation is given by

$$\begin{array}{ll} \min & \sum_i h\left(\mathbf{X}_{\mathcal{V}_i}\right) \\ \text{s.t.} & \mathbf{X}_{\mathcal{V}^i_{i\cap j}} = \mathbf{X}_{\mathcal{V}^j_{i\cap j}}, \forall (i,j) \in \mathcal{E}, \end{array} \tag{1}$$

where $\mathbf{X}_{\mathcal{V}^i_{i\cap j}}$ is the matching matrix of $\mathcal{V}_{i\cap j}$ in $\mathcal{V}_i$, i.e. a submatrix of $\mathbf{X}_{\mathcal{V}_i}$ by picking blocks that belong to the matching graphs in $\mathcal{V}_{i\cap j}$. Each $h(\mathbf{X}_{\mathcal{V}_i})$ indicates local consistency in $\mathcal{V}_i$, and the condition that $\mathbf{X}_{\mathcal{V}^i_{i\cap j}} = \mathbf{X}_{\mathcal{V}^j_{i\cap j}}$ will guarantee that the overlapping subgraph are consistent. In such a manner, the consistency condition will propagate through the overlapping sub-graph to each component $\mathcal{V}_i$ conceptually similar to our proof of Theorem 1.

In the state-of-the-art methods of [15] and [32], the authors proposed different formulations of objective function $h(\mathbf{X}_{\mathcal{V}})$. We will use the formulation described in [32] to demonstrate our framework, because of its competent performance and superior computational speed.

As in [32], $h(\mathbf{X}_{\mathcal{V}})$ can be written as

$$\begin{array}{ll} \min & \langle \mathbf{W}_{\mathcal{V}}, \mathbf{X}_{\mathcal{V}} \rangle + \lambda \|\mathbf{X}_{\mathcal{V}}\|_* \\ \text{s.t.} & \mathbf{X}_{\mathcal{V}} \succeq 0, \\ & \mathbf{X}_{\mathcal{V}(ii)} = \mathbf{I}_{m_i}, \forall i \\ & \mathbf{X}_{\mathcal{V}(ij)} = \mathbf{X}_{\mathcal{V}(ji)}^\top, \forall i \neq j \\ & 0 \leq \mathbf{X} \leq 1 \end{array} \tag{2}$$

where $\langle \cdot, \cdot \rangle$ is the matrix inner product, $\|\cdot\|_*$ is the matrix nuclear norm, and $\mathbf{W}_{\mathcal{V}} = \alpha \mathbf{1} - \overline{\mathbf{X}}_{\mathcal{V}}$, and $\mathbf{1}$ denote the matrix whose elements are 1. The purpose of adding constant α is to impose a L_1 constraint on $\mathbf{X}_{\mathcal{V}}$ to promote sparsity. We use $\mathbf{X}_{\mathcal{V}(ij)}$ to denote the $(i,j)^{\text{th}}$ block of the block matrix $\mathbf{X}_{\mathcal{V}}$. As has been shown in [32], the constraint $\mathbf{X}_{\mathcal{V}} \succeq 0$ may be relaxed for a sufficiently large λ. Let $\mathcal{C}_i$ encode the convex set induced by the constraints for $\mathcal{V}_i$, we could then simplify the formulation of our distributed problem as

$$\begin{array}{ll} \min & \sum_i \left(\langle \mathbf{W}_{\mathcal{V}_i}, \mathbf{X}_{\mathcal{V}_i} \rangle + \lambda \|\mathbf{X}_{\mathcal{V}_i}\|_*\right) \\ \text{s.t.} & \mathbf{X}_{\mathcal{V}_i} \in \mathcal{C}_i \\ & \mathbf{X}_{\mathcal{V}^i_{i\cap j}} = \mathbf{X}_{\mathcal{V}^j_{i\cap j}}, \forall (i,j) \in \mathcal{E} \end{array} \tag{3}$$

4. Alternating minimization

4.1. Algorithms

The nuclear norm minimization in (3) can be efficiently optimized using recent results on low-rank matrix recovery, which directly enforce low-rank decompositions $\mathbf{X}_{\mathcal{V}_i} = \mathbf{A}_{\mathcal{V}_i}\mathbf{B}_{\mathcal{V}_i}^\top$ (c.f. [2, 11, 32]). Here $\mathbf{A}_{\mathcal{V}_i}$ and $\mathbf{B}_{\mathcal{V}_i}$ are latent variables. According to [23], we can write the nuclear norm as

$$\|\mathbf{X}\|_* = \min_{\mathbf{A},\mathbf{B}:\mathbf{A}\mathbf{B}^\top = \mathbf{X}} \frac{1}{2}\left(\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2\right).$$

To make the notations uncluttered, we will shorten $\mathbf{X}_{\mathcal{V}_i}$ and $\mathbf{X}^i_{\mathcal{V}_{i\cap j}}$ as $\mathbf{X}_i$ and $\mathbf{X}_{ij}$, respectively. Moreover, let $\mathbf{E}_{ij}$ denote the selection matrix to extract the part of $\mathbf{X}_i$ that belongs to the set of $\mathcal{V}_i \cap \mathcal{V}_j$, i.e. $\mathbf{X}_{ij} = \mathbf{E}_{ij}^\top \mathbf{X}_i \mathbf{E}_{ij}$. With this setup, the condition on intersection consistency becomes $\mathbf{E}_{ij}^\top \mathbf{X}_i \mathbf{E}_{ij} = \mathbf{E}_{ji}^\top \mathbf{X}_j \mathbf{E}_{ji}$.

We then finalize our formation of the problem in (1) as

$$\begin{array}{ll} \min & \sum_i \left(\langle \mathbf{W}_i, \mathbf{X}_i \rangle + \frac{\lambda}{2}\|\mathbf{A}_i\|_F^2 + \frac{\lambda}{2}\|\mathbf{B}_i\|_F^2\right) \\ \text{s.t.} & \mathbf{X}_i = \mathbf{A}_i\mathbf{B}_i^\top, \\ & \mathbf{E}_{ij}^\top \mathbf{X}_i \mathbf{E}_{ij} = \mathbf{E}_{ji}^\top \mathbf{X}_j \mathbf{E}_{ji}, \\ & \mathbf{X}_i \in \mathcal{C}_i, \end{array} \tag{4}$$

In all our experiments, we set $\alpha = 0.1$, $\lambda = 50$, $\mu = 64$, and $\beta = 1$.

We apply ADMM to solve (4). The solver is summarized in Algorithm 2. In particular, $\mathbf{Y}_i$ and $\mathbf{Z}_{ij}$ are dual variables. The constraints on $\mathbf{X}$ are handled implicitly and updated in the alternating algorithm. The ADMM algorithm updates primal variables by minimizing $\mathcal{L}$ and then applies gradient descent to update the dual variables. Moreover, $\mathbf{A}_i$ and $\mathbf{B}_i$ admit closed-form solution via solving least-squares. Moreover, $\mathbf{X}_{i0}$ is the solution to the linear equation

$$\mu\mathbf{X}_i + 2\beta \sum_j \mathbf{E}_{ij}\mathbf{E}_{ij}^\top \mathbf{X}_i \mathbf{E}_{ij}\mathbf{E}_{ij}^\top = \mu\mathbf{A}_i\mathbf{B}_i^\top - (\mathbf{W}_i + \mathbf{Y}_i) + \sum_j \mathbf{E}_{ij}(2\beta\mathbf{M}^k_{j\to i} - \mathbf{Z}_{ij} + \mathbf{Z}_{ji})\mathbf{E}_{ij}^\top.$$

Furthermore, the update on $\mathbf{X}_i$ requires a projection onto the convex set $\mathcal{C}$, $\mathcal{P}_{\mathcal{C}}(\cdot)$, i.e. $\mathcal{P}_{\mathcal{C}}(\mathbf{X}_0)$ is the solution to the

Algorithm 2: Distributed Graph Matching via ADMM

Input : Initial pairwise maps $\overline{\mathbf{X}}_i$
Output: Consistent matches $\mathbf{X}_i$
Initialize $\mathbf{A}_i$, $\mathbf{B}_i$ randomly, and set $\mathbf{Y}_i$, $\mathbf{Z}_{ij}$ to be $\mathbf{0}$
$\mathbf{W}_i = \alpha\mathbf{1} - \overline{\mathbf{X}}_i$
while *not converged* **do**
  /* inner-node update */
  $\mathbf{A}_i \leftarrow (\mathbf{X}_i + \frac{1}{\mu}\mathbf{Y}_i)\mathbf{B}_i(\mathbf{B}_i^\top\mathbf{B}_i + \frac{\lambda}{\mu}\mathbf{I})^\dagger$
  $\mathbf{B}_i \leftarrow (\mathbf{X}_i + \frac{1}{\mu}\mathbf{Y}_i)\mathbf{A}_i(\mathbf{A}_i^\top\mathbf{A}_i + \frac{\lambda}{\mu}\mathbf{I})^\dagger$
  $\mathbf{X}_i \leftarrow \mathcal{P}_{\mathcal{C}_i}(\mathbf{X}_{i0})$
  $\mathbf{Y}_i \leftarrow \mathbf{Y}_i^k + \mu\left(\mathbf{X}_i - \mathbf{A}_i\mathbf{B}_i^\top\right)$
  $\mathbf{Z}_{ij} \leftarrow \mathbf{Z}_{ij}^k + \beta\left(\mathbf{M}_{i\to j}^{k+1} - \mathbf{M}_{j\to i}^{k+1}\right)$
  /* inter-node information exchange */
  node j send $\mathbf{M}_{j\to i} = \mathbf{E}_{ji}^\top\mathbf{X}_j\mathbf{E}_{ji}$ to node i
end
Round $\mathbf{X}_i$ with a threshold of 0.5.

problem

$$\min_{\mathbf{X}\in\mathcal{C}} \|\mathbf{X} - \mathbf{X}_0\|_F^2.$$

This is essentially a linear programming problem, and can be solved efficiently. We refer to the supplementary material for a derivation.

The key point in Algorithm 2 is the separation of inner-node update and inter-node information exchange. It can be seen that all the matrix computations are done in each node separately, which indicates a distributed computation. While after each iteration, adjacent nodes will need to exchange information by passing messages. Namely, for node $\mathcal{V}_j$ and all its neighboring nodes $\mathcal{V}_i$, a message will be sent in the form of $\mathbf{M}_{j\to i} = \mathbf{E}_{ji}^\top\mathbf{X}_j\mathbf{E}_{ji}$. Note that there is no overhead on generating these messages. Recall that $\mathbf{E}_{ji}$ is just a sub-block extraction matrix, and the way $\mathbf{M}_{j\to i}$ is computed is simply by extracting the sub-block of the matching matrix $\mathbf{X}_j$ that belongs to the intersection $\mathcal{V}_i \cap \mathcal{V}_j$. Since all the computations are indeed done on each node individually, the proposed algorithm is essentially completely distributed, and the only add-on is a syncing stage.

4.2. Complexity

The computational complexity of Algorithm 2 is dominated by matrix multiplication. In our approach, the per-iteration complexity is controlled by the leading node in $\mathcal{G}$, i.e. $O\left(\max_i(\sum_{H_j\in\mathcal{V}_i} m_j)^2 \max_i(|\mathcal{V}_i|)\right)$, where $|\mathcal{V}_i|$ is the total number of distinct entities among all the objects in $\mathcal{V}_i$. In contrast, the per-iteration complexity of [4] and [32] is $O((\sum_i m_i)^3)$ and $O((\sum_i m_i)^2 m)$ respectively. Furthermore, in our experiments, we found the total number of iterations to converge for our method is comparable to that of [32].

5. Experiments

5.1. Simulation

In this section, we perform experimental evaluation using synthetic datasets.

We followed the same experimental setup as in [4, 32]. Given an optimized matching matrix $\mathbf{X}^*$ and the ground truth mapping $\mathbf{X}_g$, we access the quality of $\mathbf{X}^*$ by measuring the intersection over union (or IOU) score:

$$1 - \frac{|\tau(\mathbf{X}^*) \cap \tau(\mathbf{X}_g)|}{|\tau(\mathbf{X}^*) \cup \tau(\mathbf{X}_g)|}$$

where $\tau(\cdot)$ denotes the mapping induced from the matching matrix, and $|\cdot|$ denotes the set size. Note in our distributed setting, we could only partially recover $\mathbf{X}^*$ given $\mathbf{X}_i^*$. Therefore, our ground truth setting is also different from [4, 32] in this regard.

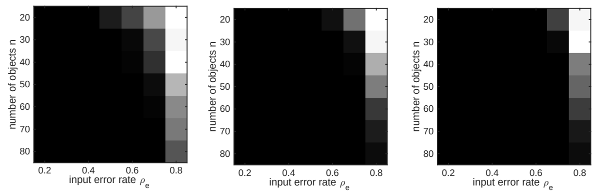
5.1.1 Matching Errors

In the first experiment, we aimed to evaluate the matching performance between our algorithm, DMatch, and the global algorithm, MatchALS, as in [32]. We considered the following model to generate the testing examples. The total number of graphs is denoted by n. The size of the universe is fixed at $r = 20$ points. In each graph, a point is randomly observed with a probability ρ_0. We simulated error corruption by randomly removing true mapping and adding false ones with a corruption rate ρ_e.

We considered two settings in our experiments. In the first setting, we constructed our cover graph by making a sparse three way tree. This was done by randomly selecting a subset of $\mathcal{V}$ as a common intersection $\mathcal{V}_c$ and then split the rest evenly into the three cover nodes $\mathcal{V}_1', \mathcal{V}_2', \mathcal{V}_3'$. As a consequence, each cover node is equal to $\mathcal{V}_i = \mathcal{V}_c \cup \mathcal{V}_i'$. In the second setting, we increased the overlap density by circularly adding one more split to each cover, i.e. $\mathcal{V}_i = \mathcal{V}_c \cup \mathcal{V}_i' \cup \mathcal{V}_{i+1}'$. We compared DMatch to MatchALS by varying the parameters ρ_0, ρ_e, and n. For both algorithms, we used $m = 2r$ and $\lambda = 50$.

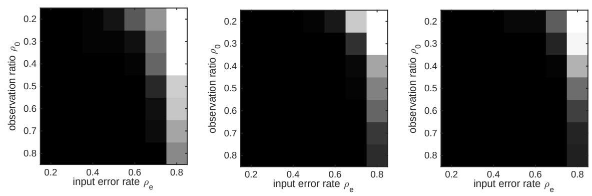
Figure 3 shows matching errors under various configurations, for both DMatch and MatchALS. In general, lowering input error and increasing observation ratio or increasing the total number of objects will improve the matching performance, i.e. with a lower matching error. In addition, we can see that increasing the overlap between cover nodes would have a positive impact on the recovery (comparison between Figure 3(a) and Figure 3(b)).

Furthermore, in a comparison between Figure 3(b) and Figure 3(c), we can see that when the cover is dense enough, i.e. the size of the overlaps are sufficiently large, the matching error would approach that of MatchALS, which is the global recovery.



DMatch-sparse DMatch-dense MatchALS

(a) number of objects n + input error rate ρ_e



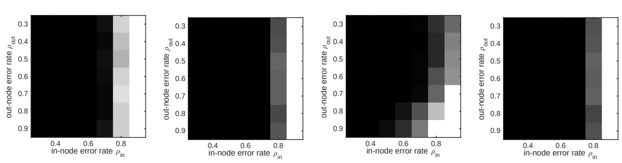
DMatch-sparse DMatch-dense MatchALS

(b) observation ratio ρ_0 + input error rate ρ_e

Figure 3: Matching error comparison. Darker color means lower matching error. The fixed parameter is set to $\rho_0 = 0.6$ and $n = 50$ respectively.

5.1.2 Graph Covers

In the second experiment, we aimed to understand more on the effects of graph covers. Specifically, we construct a ground truth graph cover by selecting a sparse cover as in Section 5.1.1. For every pair of graphs within the same cover node, we set the error rate to be ρ_{in}, and for every pair between different cover node, we set the error rate to be ρ_{out}. The experiment is then conducted by comparing DMatch to 1) using the ground truth cover and 2) using a randomly constructed cover.



sparse dense sparse dense

(a) ground truth cover (b) randomly constructed cover

Figure 4: Effects on recovery from the construction of graph cover. Darker color means lower matching error. The fixed parameter is set to $\rho_0 = 0.6$ and $n = 50$.

Figure 4 shows the experimental results. In Figure 4(a), we can see that using the ground truth cover, the matching performance does not depend much on ρ_{out}, because we explicitly disregarded any information from ρ_{out}. While on the randomly constructed cover, the two error rates are mixed up. Specifically, the results favor more on small ρ_{in} and small ρ_{out} at the same time. This situation, however, starts to change when the cover becomes denser, i.e. the increasing overlap between cover nodes. The dependency on ρ_{out} disappears as shown in Figure 4(b). One potential explanation is that with the increasing overlap between cover nodes, the portion of out-node pairs become smaller, as well as the resultant portion of induced error from them. As a consequence, the mixed error rate is dominated by ρ_{in}. In addition, we could also see that with denser cover, the algorithm is more error tolerant. This comes with a trade-off that on average the size of each cover becomes bigger and the computational cost also increases.

5.2. Real Experiments

5.2.1 CMU House Sequence

In this part of the experiment, we intend to test the scalability of our distributed algorithm. We use the CMU House sequence[1] as the testing images. This sequence has been widely used to test different graph matching algorithms. It consists of 110 frames, and there are 30 feature points labeled consistently across all frames. We used the geometry based constraint in pairwise matching as is done in [14]. To construct a valid cover complex $\mathcal{K}$, we first computed all pairwise matches and for each match, the result is encoded using a binary matrix. We then built a fully connected matching quality graph, where each vertex represents an image, and the edge weight represents the matching score associated with each image pair. Since we knew the sequence was roughly generated by moving a camera in a single dimension, we computed the Laplacian embedding of the vertices onto 1 dimensional space using Fielder vector, and then we applied Algorithm 1 to build the cover complex. We compared our DMatch algorithm with the global methods, MatchALS [32], MatchLift [4], Spectral [22], and the distributed method described in [17]. Besides running global algorithms on fully connected map graph, we also ran them on the sparse map graph induced by our cover complex, marked with different K values. The algorithm was implemented in a single laptop with Intel Core i7 2.0GHz CPU and 16GB DDR3 Memory. We measured the time used in each cover node separately and then took the max as the total computational time per iteration, where we assumed the cost for messages passed between adjacent cover nodes was negligible.

Table 1 shows the matching accuracy, timing and iterations used in these algorithms. In our experiments, Spectral method appeared to be the fastest method among all the methods. On the other hand, it has the largest reported error. The consensus algorithm is not error-driven, hence we used a preset 150 iterations to match the lowest number of iterations among all experiments. In general, distributed algorithms used less iterations to converge, and achieved at

[1] http://vasc.ri.cmu.edu//idb/html/motion/house/index.html

	Error Rate	Iterations	Time
Original	0.1445	-	-
MatchALS	0.1031	266	98.8
MatchLift	0.1027	1000	3791.1
Spectral	0.1277	-	**0.6**
MatchALS ($K = 4$)	0.0161	380	103.5
MatchALS ($K = 6$)	0.0648	1000	268.2
MatchLift ($K = 4$)	**0**	1000	4066.1
MatchLift ($K = 6$)	**0**	1000	3972.3
DMatch ($K = 4$)	**0**	203	28.9
DMatch ($K = 6$)	**0**	**150**	7.6
Consensus ($K = 4$)	**0**	150	28.3
Consensus ($K = 6$)	0.0071	150	18.7

Table 1: The error rate and the total computational time (seconds) on CMU House sequence.

least an order of magnitude speed-up compared with global methods, while maintaining an error rate of **0** (almost 0 for Consensus algorithm with $K = 6$). It can also be seen that for distributed algorithms once we increased the number of clusters from $K = 4$ to $K = 6$, both the number of iterations and total computational time decreased. This, on the other hand, proved our complexity analysis, since increasing the number of clusters would in general reduce the number of vertices in each cluster. Although both DMatch and Consensus algorithms achieved similar results in terms of accuracy and time, the latter requires knowledge of the number of universal entities and has limitations dealing with partial matches. Another interesting observation is after convergence, the error of MatchALS on the sparse map graph induced by our cover complex is reduced, when compared with that on the fully connected map graph. One explanation is that the graph cover structure grouped together images that have high pairwise matching quality and explicitly disregard any pairwise matches that are of low quality (covers are joint normal), and as a consequence, it is robust against noisy pair-wise matches. In addition, we also extended the experiments for $K = 8, 10$ and found the running time was not reduced significantly, in comparison with that from $K = 4$ to $K = 6$. Since we need to have a reasonable amount of overlaps between clusters to pass matching information around, increasing the number of clusters does not necessarily reduce the maximum size of the clusters.

5.2.2 Graffiti datasets

In this experiment, we followed the procedure described in [32]. We used the benchmark datasets from Graffiti datasets [2]. In each dataset, there are 6 images of a scene with various image transformations, including viewpoint change, blurring, and illumination variation etc.

To construct an affinity score matrix $\overline{\mathbf{X}}$, we employed the same procedure as in [32] for comparison purpose. We first detected 1000 affine covariant feature [19] points in each image of the dataset and computed their SIFT [18] descriptors using VLFeat library [25]. The affinity scores were computed as the inner product between every pair of detected feature points on each pair of images. We excluded apparent mismatches by keeping only affinity scores that are above the threshold 0.7. Furthermore, any potential matches that are indistinguishable was removed, i.e. if the first and the second top matches were below the ratio threshold 1.1, the candidate point was removed. Finally, any feature point that has only one candidate match in the dataset was also excluded.

In a comparison, to construct our cover graph, we first built a matching quality graph, using the matching score as the edge weight and used the Fiedler vector of the graph laplacian as the embedding and applied Algorithm 1 to build the cover complex.

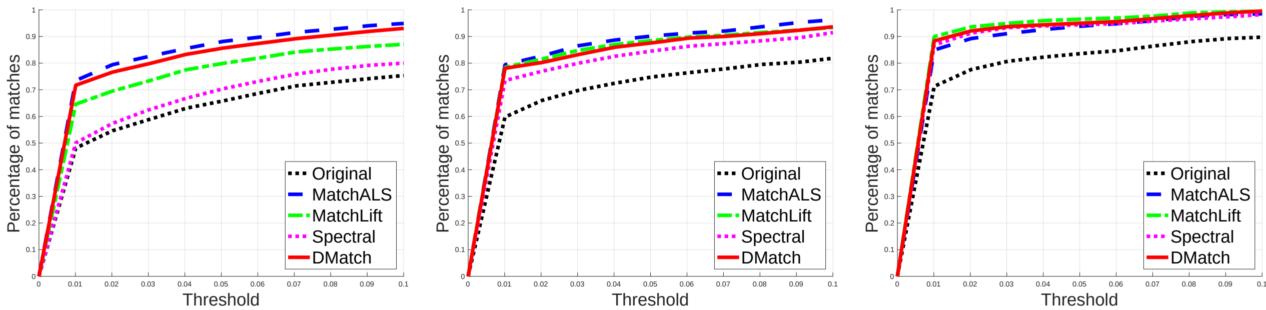
To evaluate the performance, we used the ground truth homography matrix given in the dataset, and adopted the procedure used in [4]. For a testing point, we calculated the true correspondence using homography and compared with the matched correspondence. If they were within a predefined distance threshold, we deemed the matching is correct, and otherwise, wrong. Then we swept along the threshold dimension to draw an error curve that is dependent on the threshold chosen.

We tested our DMatch algorithm against MatchALS [32], MatchLift [4], Spectral [22] and the original pairwise matchings. We ignored the Consensus algorithm [17] as it cannot explicitly handle partial matches. Figure 5 shows the curve for three datasets, Graffiti, Bikes, and Leuven. Note that DMatch will not give a full pairwise matching between images, instead, we only have a matching when the two images belong to the same cover node. Therefore, we computed the one-hop composite match between image pairs across different cover nodes[3]. From the performance curve, we can see that our DMatch performs very similar to the best global methods in all datasets as shown in [32]. In another word, DMatch achieved performance gains without loss of matching quality.

In Figure 6, we show the example matches between the first and the fourth image for each dataset. The bottom match is DMatch, the middle is MatchALS and the top one is the original pairwise map. Clearly, our matching shows at least as good as the results of MatchALS, where both cor-

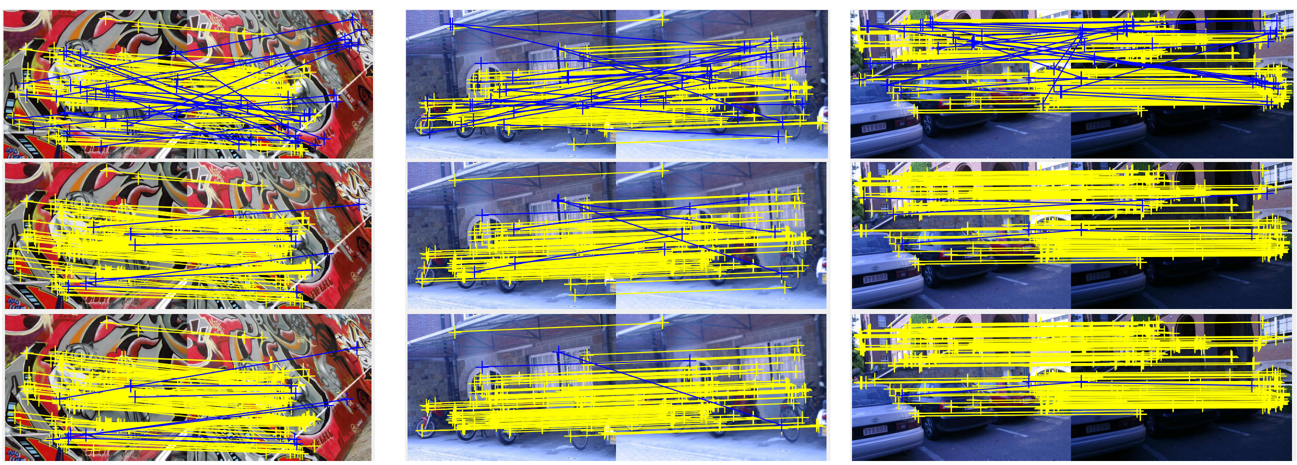
[2] http://www.robots.ox.ac.uk/~vgg/data/data-aff.html

[3] For each image i and j not in the same cover node, we loop through all $k \neq i, j$ and accumulate the composite matchings from $i \to k$ and $k \to j$.



(a) Graffiti (b) Bikes (c) Leuven

Figure 5: The performance curve on Graffiti, Bikes and Leuven datasets. The y-axis is the correct match ration and the x-axis is the threshold value over the image width. We compare DMatch (red solid) with MatchALS [32] (blue dashed), MatchLift [15], Spectral [22], and original pairwise matching (black dotted).



(a) Graffiti (b) Bikes (c) Leuven

Figure 6: Example of matching results. The bottom one is from DMatch, the middle one is from MatchALS, and the top one is from original pairwise matching respectively. Yellow lines encode the correct matches, while blue lines are for wrong ones.

rected mismatches (reduced blue lines) and increased correct matches (denser yellow lines).

In our implementation we notice that the total number of iterations to converge for both DMatch and MatchALS are roughly the same (around 60 iterations).

6. Conclusion

In this paper, we introduced a scalable framework for establishing consistent matches across multiple graphs in a distributed manner. We showed how to use our framework to extend state-of-the-art global methods. By running an iterative optimization algorithm locally and exchange information in every iteration, our framework would achieve local and global consistent matching at the same time. Furthermore, we theoretically proved the sufficient conditions under which locally consistent matching would guarantee global consistency. In our experiments, we showed that in practice, the assumptions and the conditions in our theorem could be relaxed without sacrificing performance. In addition, our proposed distributed framework achieved order of magnitude improvements in speed. We believe this is a very important first step for large scale exploration of images for object matching as well as building 3D object models from crowd-sourced collections. Future work includes matching large collection of different deformable objects that have high similarity and enough variance, e.g. a collection of different dogs or cats.

Acknowledgements: The authors wish to thank the support of NSF grants DMS-1546206 and DMS-1700234, a Google focused research award and a gift from Adobe Research.

References

[1] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. *CVPR*, 2000. 1

[2] R. Cabral, F. De la Torre, J. P. Costeira, and A. Bernardino. Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition. *ICCV*, 2013. 4

[3] X. Chen, A. Shrivastava, and A. Gupta. Enriching visual knowledge bases via object discovery and segmentation. *CVPR*, pages 2035–2042, 2014. 1

[4] Y. Chen, L. Guibas, and Q. Huang. Near-optimal joint object matching via convex relaxation. *ICML*, 2014. 1, 2, 3, 5, 6, 7

[5] L. Cosmo, E. Rodola, A. Albarelli, F. Memoli, and D. Cremers. Consistent partial matching of shape collections via sparse modeling. *CGF*, 2016. 1

[6] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *NIPS'06*, pages 313–320, 2006. 1

[7] Y. Dai, H. Li, and M. He. A simple prior-free method for non-rigid structure-from-motion factorization. *CVPR*, 2012. 1

[8] A. Egozi, Y. Keller, and H. Guterman. A probabilistic approach to spectral graph matching. *IEEE Transactions on PAMI*, 99(PrePrints), 2012. 1

[9] D. Emms, R. C. Wilson, and E. R. Hancock. Graph matching using the interference of continuous-time quantum walks. *Pattern Recogn.*, 42(5):985–1002, May 2009. 1

[10] M. Gori, M. Maggini, and L. Sarti. Exact and approximate graph matching using random walks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7):1100–1111, July 2005. 1

[11] T. Hastie, R. Mazumder, L. J, and Z. R. Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research*, pages 3367–3402, Jan 2015. 4

[12] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002. 3

[13] N. Hu, R. Rustamov, and L. Guibas. Graph matching with anchor nodes: A learning approach. *CVPR*, pages 2906 – 2913, 2013. 1

[14] N. Hu, R. Rustamov, and L. Guibas. Stable and informative spectral signatures for graph matching. *CVPR*, pages 2313 – 2320, 2014. 1, 6

[15] Q.-X. Huang and L. J. Guibas. Consistent shape maps via semidefinite programming. *SGP*, 32(5):177–186, 2013. 1, 2, 3, 4, 8

[16] J. Lee, M. Cho, and K. M. Lee. Hyper-graph matching via reweighted random walks. *CVPR*, pages 1633 – 1640, 2011. 1

[17] S. Leonardos, X. Zhou, and K. Daniilidis. Distributed consistent data association. *ICAR*, 2017. 2, 6, 7

[18] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91?110, 2004. 7

[19] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 65(1-2):43?72, 2005. 7

[20] A. Nguyen, M. Ben-Chen, K. Welnicka, Y. Ye, and L. Guibas. An optimization approach to improving collections of shape maps. *Computer Graphics Forum*, 30(5):1481–1491, 2011. 1

[21] D. Pachauri, R. Kondor, G. Sargur, and V. Singh. Permutation diffusion maps (pdm) with application to the image association problem in computer vision. *NIPS*, 2014. 1

[22] D. Pachauri, R. Kondor, and V. Singh. Solving the multi-way matching problem by permutation synchronization. *NIPS*, 2013. 1, 2, 6, 7, 8

[23] B. Recht, M. Fazel, and P. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010. 4

[24] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. PAMI*, 10(5):695–703, 1988. 1

[25] A. Vedaldi and B. Fulkerson. *VLFeat: An open and portable library of computer vision algorithms*, 2008. http://www.vlfeat.org/. 7

[26] R. C. Wilson and P. Zhu. A study of graph spectra for comparing graphs and trees. *Pattern Recogn.*, 41(9):2833–2841, Sept. 2008. 1

[27] J. Yan, M. Cho, H. Zha, X. Yang, and S. Chu. Multi-graph matching via affinity optimization with graduated consistency regularization. *IEEE Trans. PAMI*, 2015. 1

[28] J. Yan, J. Wang, H. Zha, X. Yang, and S. Chu. Consistency-driven alternating optimization for multigraph matching: A unified approach. *IEEE Trans Image Processing*, 24(3):994–1009, 2015. 2

[29] J. Yan, H. Xu, H. Zha, X. Yang, H. Liu, and S. Chu. A matrix decomposition perspective to multiple graph matching. *ICCV*, 2015. 2

[30] C. Zach, M. Klopschitz, and M. Pollefeys. Disambiguating visual relations using loop constraints. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1426–1433, 2010. 1

[31] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. *CVPR*, 2008. 1

[32] X. Zhou, M. Zhu, and K. Daniilidis. Multi-image matching via fast alternating minimization. *International Conference on Computer Vision (ICCV)*, 2015. 1, 2, 4, 5, 6, 7, 8

[33] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33:249–274, 2005. 3