Factorized Graph Matching

Feng Zhou **Fernando De la Torre**
Robotics Institute, Carnegie Mellon University
www.f-zhou.com ftorre@cs.cmu.edu

Abstract

Graph matching plays a central role in solving correspondence problems in computer vision. Graph matching problems that incorporate pair-wise constraints can be cast as a quadratic assignment problem (QAP). Unfortunately, QAP is NP-hard and many algorithms have been proposed to solve different relaxations. This paper presents factorized graph matching (FGM), a novel framework for interpreting and optimizing graph matching problems. In this work we show that the affinity matrix can be factorized as a Kronecker product of smaller matrices. There are three main benefits of using this factorization in graph matching: (1) There is no need to compute the costly (in space and time) pair-wise affinity matrix; (2) The factorization provides a taxonomy for graph matching and reveals the connection among several methods; (3) Using the factorization we derive a new approximation of the original problem that improves state-of-the-art algorithms in graph matching. Experimental results in synthetic and real databases illustrate the benefits of FGM. The code is available at `http://humansensing.cs.cmu.edu/fgm`.

1. Introduction

Graph matching plays a central role in solving many correspondence problems in computer vision such as shape matching [4], object categorization [15], feature tracking [23, 33], symmetry analysis [10, 22] and action recognition [6, 20]. Mathematically, pair-wise graph matching is formulated as the quadratic assignment problem (QAP) [27]. Unlike the linear assignment problem, which can be efficiently solved with the Hungarian algorithm [7], QAP is known to be NP-hard [19] and an exact optimal algorithm can only work for very small graphs. Therefore, the main body of research in QAP has focused on devising more accurate and faster algorithms to solve it approximately.

Although extensive research has been done for decades, graph matching is still a challenging problem mainly due to two reasons: (1) In general, the objective function is non-convex and prone to local minima; (2) The constraints that the solution has to satisfy are combinatorial. While there are a number of papers [11, 13, 21, 24, 37, 40] addressing the second issue, fewer papers [25, 39] have investigated the first issue.

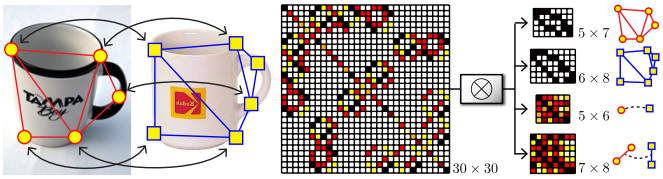

Figure 1. Matching two coffee mugs with 5 and 6 features respectively. The original pair-wise affinity matrix is of size 30×30. Our algorithm exploits the particular structure of the affinity matrix and is able to factorize it as a Kronecker product of four smaller matrices. The top two matrices of size 5×7 and 6×8 represent the structure of the graphs in each image. The lower two matrices encode the affinities for nodes (5×6) and edges (7×8).

In this paper, we show that for most pair-wise graph matching problems the affinity matrix can be factorized as a Kronecker product of smaller matrices. Based on this fact, we proposed factorized graph matching (FGM), a novel framework for interpreting and optimizing graph matching problems. The benefits of our approach are three fold: (1) It avoids the computation of the cumbersome affinity matrix and hence potentially allows for a more efficient implementation, especially for large graphs; (2) Many graph matching methods can be understood as an instance of this factorization. This allows understanding commonalities and differences among many pair-wise graph matching problems; (3) The factorization leads to a new approximation of the graph matching problem that improves state-of-the-art approaches. Fig. 1 illustrates an example of matching two coffee mugs using FGM. Note that FGM factorizes the large 30×30 affinity matrix into four smaller ones.

2. Previous work

This section reviews the problem formulation of graph matching and discusses recent advances in solving the QAP in graph matching.

2.1. Problem formulation of graph matching

We denote (see notation[1]) a graph by $\mathcal{G} = \{\mathbf{P}, \mathbf{Q}, \mathbf{G}\}$, where $\mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_n] \in \mathbb{R}^{d_p \times n}$ and $\mathbf{Q} =$

[1]Bold capital letters denote a matrix $\mathbf{X}$, bold lower-case letters a column vector $\mathbf{x}$. $\mathbf{x}_i$ represents the i^{th} column of the matrix $\mathbf{X}$. x_{ij} denotes

$[\mathbf{q}_1, \cdots, \mathbf{q}_m] \in \mathbb{R}^{d_q \times m}$ are the feature matrices computed for nodes and edges[2] respectively. The topology of $\mathcal{G}$ is specified by a node-edge incidence matrix $\mathbf{G} \in \{0,1\}^{n \times m}$, where $g_{ic} = g_{jc} = 1$ if the i^{th} and j^{th} nodes are connected by the c^{th} edge, and zero otherwise. For instance, Fig. 2a shows a pair of synthetic graphs and Fig. 2cd illustrate their incidence matrices.

Suppose that we are given a pair of graphs, $\mathcal{G}_1 = \{\mathbf{P}_1, \mathbf{Q}_1, \mathbf{G}_1\}$ and $\mathcal{G}_2 = \{\mathbf{P}_2, \mathbf{Q}_2, \mathbf{G}_2\}$. We compute two affinity matrices, $\mathbf{K}_p \in \mathbb{R}^{n_1 \times n_2}$ and $\mathbf{K}_q \in \mathbb{R}^{m_1 \times m_2}$, for measuring the similarity of each node and edge pair respectively. More specifically, $\kappa^p_{i_1 i_2} = \phi_p(\mathbf{p}^1_{i_1}, \mathbf{p}^2_{i_2})$ measures the similarity between the i_1^{th} node of $\mathcal{G}_1$ and the i_2^{th} node of $\mathcal{G}_2$, and $\kappa^q_{c_1 c_2} = \phi_q(\mathbf{q}^1_{c_1}, \mathbf{q}^2_{c_2})$ measures the similarity between the c_1^{th} edge of $\mathcal{G}_1$ and the c_2^{th} edge of $\mathcal{G}_2$. The problem of graph matching consists in finding a correspondence between the nodes of $\mathcal{G}_1$ and $\mathcal{G}_2$ that maximizes the following score of global consistency:

$$J_{gm}(\mathbf{X}) = \sum_{i_1 i_2} x_{i_1 i_2} \kappa^p_{i_1 i_2} + \sum_{\substack{i_1 \neq i_2, j_1 \neq j_2 \\ g^1_{i_1 c_1} g^1_{j_1 c_1} = 1 \\ g^2_{i_2 c_2} g^2_{j_2 c_2} = 1}} x_{i_1 i_2} x_{j_1 j_2} \kappa^q_{c_1 c_2},$$

where $\mathbf{X} \in \{0,1\}^{n_1 \times n_2}$ denotes the node correspondence, *i.e.*, $x_{i_1 i_2} = 1$ if the i_1^{th} node of $\mathcal{G}_1$ corresponds to the i_2^{th} node of $\mathcal{G}_2$. In most cases, $\mathbf{X}$ is constrained to be a one-to-one matching, *i.e.*, $\mathbf{X}\mathbf{1}_{n_2} \leq \mathbf{1}_{n_1}$ and $\mathbf{X}^T\mathbf{1}_{n_1} \leq \mathbf{1}_{n_2}$.

It is more convenient to write $J_{gm}(\mathbf{X})$ in a quadratic form, $\mathbf{x}^T\mathbf{K}\mathbf{x}$, where $\mathbf{x} = \mathrm{vec}(\mathbf{X}) \in \{0,1\}^{n_1 n_2}$ is an indicator vector and $\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ is computed as follows:

$$\kappa_{i_1 i_2 j_1 j_2} = \begin{cases} \kappa^p_{i_1 i_2}, & \text{if } \; i_1 = j_1 \text{ and } i_2 = j_2, \\ \kappa^q_{c_1 c_2}, & \text{if } \; i_1 \neq j_1 \text{ and } i_2 \neq j_2 \text{ and} \\ & g^1_{i_1 c_1} g^1_{j_1 c_1} g^2_{i_2 c_2} g^2_{j_2 c_2} = 1, \\ 0, & \text{otherwise.} \end{cases}$$

For instance, Fig. 2e-g illustrates the composition of the affinity matrices. With these notations, the goal of graph matching is to optimize the following QAP:

$$\max_{\mathbf{x}} \quad \mathbf{x}^T\mathbf{K}\mathbf{x}, \quad \text{s.t. } \mathbf{A}\mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{x} \in \{0,1\}^{n_1 n_2}, \quad (1)$$

$$\text{where} \quad \mathbf{A} = \begin{bmatrix} \mathbf{1}^T_{n_2} \otimes \mathbf{I}_{n_1} \\ \mathbf{I}_{n_2} \otimes \mathbf{1}^T_{n_1} \end{bmatrix} \text{ and } \mathbf{b} = \mathbf{1}_{n_1 + n_2}.$$

the scalar in the i^{th} row and j^{th} column of the matrix $\mathbf{X}$. All non-bold letters represent scalars. $\mathbf{1}_{m \times n}, \mathbf{0}_{m \times n} \in \mathbb{R}^{m \times n}$ are matrices of ones and zeros. $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ is an identity matrix. $\|\mathbf{x}\|_p = \sqrt[p]{\sum |x_i|^p}$ denotes the p-norm. $\|\mathbf{X}\|_F^2 = \mathrm{tr}(\mathbf{X}^T\mathbf{X})$ designates the Frobenious norm. $\mathrm{vec}(\mathbf{X})$ denotes the vectorization of matrix $\mathbf{X}$. $\mathrm{diag}(\mathbf{x})$ is a diagonal matrix whose diagonal elements are $\mathbf{x}$. $\mathbf{X} \circ \mathbf{Y}$ and $\mathbf{X} \otimes \mathbf{Y}$ are the Hadamard and Kronecker products of matrices. $\{i : j\}$ lists the integers, $\{i, i+1, \cdots, j-1, j\}$. $\mathrm{eig}(\mathbf{X})$ computes the leading eigen-vector of $\mathbf{X}$.

[2] In general, the edge feature can be asymmetrical, *i.e.*, the feature used for edge $\overrightarrow{ij}$ is different from $\overrightarrow{ji}$. However, the symmetrical edge feature can express a wide range of graph matching problems. For instance, the pairwise distance and the absolute angle from the horizontal line both belong to this class of edge feature.

2.2. Advances in graph matching

Over the past three decades, a myriad of approximations to solve the QAP in graph matching have been proposed in computer vision and machine learning (see [12, 30] for a survey). These methods can be broadly categorized in two types based on the objective to be maximized: $\mathrm{tr}(\mathbf{A}_1\mathbf{X}\mathbf{A}_2\mathbf{X}^T)$ and $\mathbf{x}^T\mathbf{K}\mathbf{x}$.

The first case corresponds to maximizing a trace-form objective function, $\mathrm{tr}(\mathbf{A}_1\mathbf{X}\mathbf{A}_2\mathbf{X}^T)$, where $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{R}^{n \times n}$ are the weighted adjacency matrices of the graphs and $\mathbf{X} \in \{0,1\}^{n \times n}$ is a permutation matrix. In the literature of operation research [27], this is known as Koopmans-Beckmann's QAP, which is a particular case of Lawler's QAP maximizing $\mathbf{x}^T\mathbf{K}\mathbf{x}$ when $\mathbf{K} = \mathbf{A}_2 \otimes \mathbf{A}_1$. In the past two decades, various continuous relaxations have been proposed to solve this type of problems. Umeyama [36] proposed the first spectral algorithm by computing the eigenvectors of the adjacency matrices. Almohamad and Duffuaa [3] proposed to optimize an l_1-norm objective function by linear programming. The most related work to ours is the one from Zaslavskiy *et al*. [39], in which a convex-concave approach was proposed to estimate the correspondence in an iterative manner. Despite its successfulness for matching characters and other visual objects with relatively simple structure, the graphs used by these methods still lack flexibility to match complex structures encountered in realistic computer vision problems.

In the more general case, the problem is formulated as the maximization of a quadratic cost $\mathbf{x}^T\mathbf{K}\mathbf{x}$, where $\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ encodes the pair-wise similarity between nodes and edges. In the past decade, much effort has been devoted to the development of approximate methods to solve the more general QAP. Gold and Rangarajan [21] proposed the graduated assignment algorithm to iteratively solve a series of linear approximations of the cost function using Taylor expansions. Leordeanu and Hebert [24] proposed an efficient approximation using an spectral relaxation. Cour *et al*. [13] presented a more general scheme that incorporates affine constraints in the spectral relaxation, thereby obtaining better approximation of the original problem. Van Wyk and van Wyk [37] proposed to iteratively project the approximate correspondence matrix onto the convex domain of the desired integer constraints. Torresani *et al*. [35] designed a complex objective function which can be efficiently optimized by dual decomposition. As a general tool for approximating combinatorial problems, semi-definite programming [34, 31] was also used to approximate graph matching. Recently, Leordeanu *et al*. [25] proposed an integer projection algorithm to optimize the objective function in an integer domain. In addition to optimization-based work, probabilistic frameworks [11, 40] were shown to be useful for interpreting and solving graph matching problems. In our work, we concentrate on solv-

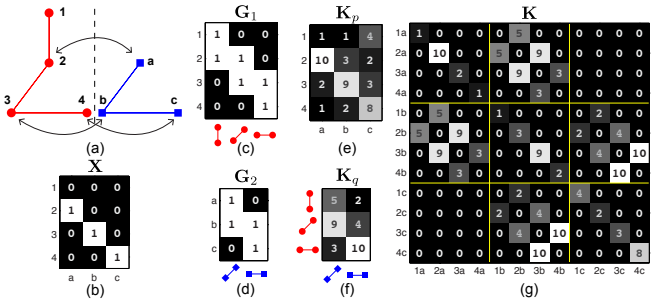

Figure 2. Example of graph matching and related matrices. (a) Two synthetic graphs. (b) The correspondence matrix $\mathbf{X}$. (c) The 1^{st} graph's incidence matrix $\mathbf{G}_1$. (d) The 2^{nd} graph's incidence matrix $\mathbf{G}_2$. (e) The node affinity matrix $\mathbf{K}_p$. (f) The edge affinity matrix $\mathbf{K}_q$. (g) The global affinity matrix $\mathbf{K}$.

ing the most general type of graph matching problem using optimization techniques.

3. Factorized graph matching (FGM)

It is well known that the QAP (Eq. 1) is one of the most difficult combinatorial optimization problems. In general, instances of size $n > 20$ cannot be exactly solved in practical time. Many methods have been proposed to compute an approximate solution. In particular, most efforts focus on maximizing $J_{gm}(\mathbf{X})$ by relaxing the binary constraints. For instance, a popular relaxation is to constrain $\mathbf{X}$ as a doubly stochastic matrix [11, 21, 37, 40], which is the convex hull of permutation matrices. Though the constraint can be relaxed to be convex, we still need to tackle a hard non-convex quadratic programming since $\mathbf{K}$ is not necessarily negative definite.

To be able to derive a better optimization scheme for addressing the non-convex issue, this section exploits the underlying structure of $\mathbf{K}$. In particular, $\mathbf{K}$ can be factorized into smaller matrices. With this new factorization of $\mathbf{K}$, many graph matching methods can be re-interpreted in a coherent manner. Consider the synthetic graph shown in Fig. 2. Our main intuition relies on two observations. First, the large affinity matrix, $\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ is divided into n_2-by-n_2 smaller blocks $\mathbf{K}_{ij} \in \mathbb{R}^{n_1 \times n_1}$. Some of $\mathbf{K}_{ij}$s contain only zero-value elements and their positions are indexed by $\mathbf{G}_2\mathbf{G}_2^T$, *i.e.*, $\mathbf{K}_{ij} = \mathbf{0}_{n_1 \times n_1}$ if $[\mathbf{G}_2\mathbf{G}_2^T]_{ij} = 0$. Second, all the non-diagonal elements of $\mathbf{K}_{ij}$ can be computed as $\mathbf{G}_1 \operatorname{diag}(\mathbf{k}_c^q)\mathbf{G}_1^T$, where $c \in \{1 : m_2\}$ is the index of the edge connecting the i^{th} and j^{th} nodes of $\mathcal{G}_2$ (*i.e.*, $g_{ic}^2 = g_{jc}^2 = 1$). Based on these two observations, and after some linear algebra, it can be shown that $\mathbf{K}$ can be factorized as:

$$\mathbf{K} = (\mathbf{H}_2 \otimes \mathbf{H}_1) \operatorname{diag}(\operatorname{vec}(\mathbf{L}))(\mathbf{H}_2 \otimes \mathbf{H}_1)^T, \tag{2}$$

where $\mathbf{H}_1 = [\mathbf{G}_1, \mathbf{I}_{n_1}] \in \{0,1\}^{n_1 \times (m_1 + n_1)}$,

$$\mathbf{H}_2 = [\mathbf{G}_2, \mathbf{I}_{n_2}] \in \{0,1\}^{n_2 \times (m_2 + n_2)},$$

$$\mathbf{L} = \begin{bmatrix} \mathbf{K}_q & -\mathbf{K}_q\mathbf{G}_2^T \\ -\mathbf{G}_1\mathbf{K}_q & \mathbf{G}_1\mathbf{K}_q\mathbf{G}_2^T + \mathbf{K}_p \end{bmatrix} \in \mathbb{R}^{(m_1+n_1)\times(m_2+n_2)}.$$

Observe that this factorization decouples the graph structure ($\mathbf{H}_2 \otimes \mathbf{H}_1$) from the pairwise feature ($\mathbf{L}$). To the best of our knowledge, Eq. 2 is the first time that $\mathbf{K}$ is factorized as products of $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{K}_p$ and $\mathbf{K}_q$. As we will see in the rest of the paper, this will have important implications for our graph matching algorithm. This closed-form paves the way to approaching the graph matching problem by manipulating the smaller and denser $\mathbf{L}$ instead of the very large and sparse $\mathbf{K}$. Plugging the factorization of $\mathbf{K}$ into $J_{gm}(\mathbf{X})$ leads to an equivalent trace-form objective function:

$$\begin{aligned} J_{gm}(\mathbf{X}) &= \mathbf{x}^T(\mathbf{H}_2 \otimes \mathbf{H}_1) \operatorname{diag}(\operatorname{vec}(\mathbf{L}))(\mathbf{H}_2 \otimes \mathbf{H}_1)^T\mathbf{x} \\ &= \operatorname{vec}(\mathbf{H}_1^T\mathbf{X}\mathbf{H}_2)^T \operatorname{diag}(\operatorname{vec}(\mathbf{L})) \operatorname{vec}(\mathbf{H}_1^T\mathbf{X}\mathbf{H}_2) \\ &= \operatorname{tr}\Big(\mathbf{L}^T(\mathbf{H}_1^T\mathbf{X}\mathbf{H}_2 \circ \mathbf{H}_1^T\mathbf{X}\mathbf{H}_2)\Big). \end{aligned} \tag{3}$$

Observe that $\mathbf{L}$ can always be factorized (*e.g.*, SVD) as $\mathbf{L} = \mathbf{U}\mathbf{V}^T = \sum_{i=1}^{c} \mathbf{u}_i\mathbf{v}_i^T$, where $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_c] \in \mathbb{R}^{(n_1+m_1)\times c}$ and $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_c] \in \mathbb{R}^{(n_2+m_2)\times c}$ and $c \leq \min(n_1 + m_1, n_2 + m_2)$. Substituting it into Eq. 3 yields[3] an equivalent trace form of $J_{gm}(\mathbf{X})$:

$$\begin{aligned} J_{gm}(\mathbf{X}) &= \sum_{i=1}^{c} \operatorname{tr}\Big(\operatorname{diag}(\mathbf{u}_i)\mathbf{H}_1^T\mathbf{X}\mathbf{H}_2 \operatorname{diag}(\mathbf{v}_i)\mathbf{H}_2^T\mathbf{X}^T\mathbf{H}_1\Big) \\ &= \sum_{i=1}^{c} \operatorname{tr}(\mathbf{A}_i^1\mathbf{X}\mathbf{A}_i^2\mathbf{X}^T), \end{aligned} \tag{4}$$

where $\mathbf{A}_i^1 = \mathbf{H}_1 \operatorname{diag}(\mathbf{u}_i)\mathbf{H}_1^T$ and $\mathbf{A}_i^2 = \mathbf{H}_2 \operatorname{diag}(\mathbf{v}_i)\mathbf{H}_2^T$.

At this point, it is important to notice that Eq. 3 and Eq. 4 can represent many graph matching methods in a unified manner.

Spectral relaxation: Suppose $\mathbf{L}$ has a rank-1 structure, *i.e.*, $c = 1$ and $\mathbf{L} = \mathbf{u}\mathbf{v}^T$. Then the kernel matrix $\mathbf{K}$ can be factorized as $\mathbf{K} = \mathbf{A}_2 \otimes \mathbf{A}_1$. Therefore, the solution of spectral matching algorithm using eigen-decomposition [24] can be efficiently computed as $\operatorname{eig}(\mathbf{K}) = \operatorname{eig}(\mathbf{A}_2) \otimes \operatorname{eig}(\mathbf{A}_1)$. In addition, we can use Umeyama's spectral algorithm [36] to find the approximate solution by maximizing $\operatorname{tr}(\mathbf{A}_1\mathbf{X}\mathbf{A}_2\mathbf{X}^T)$ subject to $\mathbf{X}\mathbf{X}^T = \mathbf{I}$.

Edge matching: Observe that both $\mathbf{H}_1^T\mathbf{X}\mathbf{H}_2$ and $\mathbf{L}$ have a 2-by-2 block structure and their top-left components are $\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2$ and $\mathbf{K}_q$ respectively. Recall that $\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2 \in \{0,1\}^{m_1 \times m_2}$ encodes the correspondence between edges. Intuitively, the goal of maximizing Eq. 3 is to seek for the edge-edge correspondence matrix $\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2$ such that $\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2 \circ \mathbf{G}_1^T\mathbf{X}\mathbf{G}_2$ is as correlated as possible with $\mathbf{K}_q$. The idea of matching edges has also been used in the probabilistic matching algorithm [40], where Zass and Shashua proposed to maximize the correlation between $\mathbf{X}$ and $\mathbf{G}_1\mathbf{K}_q\mathbf{G}_2^T$.

Unified view: Eq. 4 reveals the connection between two types of graph matching problems, the less general

[3] The formula $\operatorname{tr}\Big((\mathbf{u}\mathbf{v}^T)^T(\mathbf{A} \circ \mathbf{B})\Big) = \operatorname{tr}\Big(\operatorname{diag}(\mathbf{u})\mathbf{A} \operatorname{diag}(\mathbf{v})\mathbf{B}^T\Big)$ always holds for arbitrary $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$.

one [3, 36, 39] that maximizes $\mathrm{tr}(\mathbf{A}_1\mathbf{X}\mathbf{A}_2\mathbf{X}^T)$, versus the more general one [11, 13, 21, 24, 25, 37, 40] that maximizes $\mathbf{x}^T\mathbf{K}\mathbf{x}$. In particular, maximization of $\mathbf{x}^T\mathbf{K}\mathbf{x}$ can be equivalently cast as the maximization of the sum of c traces $\mathrm{tr}(\mathbf{A}_i^1\mathbf{X}\mathbf{A}_i^2\mathbf{X}^T)$, where $\mathbf{A}_i^1$ and $\mathbf{A}_i^2$ can be interpreted as adjacency matrices. In the special case when $c = 1$, the two types of problems are equivalent.

4. Optimization for factorized graph matching

Due to its combinatorial nature, Eq. 1 is usually approached by a two-step scheme: (1) solving a contiguously relaxed problem and (2) rounding the approximate solution to a binary one. Conventional methods perform these two steps independently. As mentioned in [25, 39], however, this kind of separate treatment will inevitably cause accuracy loss, especially in the rounding step which is independent of the cost function (Eq. 1). Inspired by [29, 39], we address these two issues in a coherent manner by iteratively optimizing an interpolation of two relaxations. This new scheme has three theoretical advantages: (1) The optimization performance is initialization-free; (2) The final solution is guaranteed to converge at an integer one and therefore no rounding step is needed; (3) The iteratively updating procedure resembles the idea of numerical continuation methods [2], which have been successfully used for solving nonlinear systems of equations in decades.

4.1. A convex relaxation

In this section, we introduce a convex relaxation for Eq. 1 assuming $\mathbf{X}$ is orthogonal and using the properties of the new factorization.

Strictly speaking, the $\mathbf{X}$ satisfying the constraint in Eq. 1 is not a permutation matrix when $n_1 \neq n_2$. However, we can always slightly change the problem setting by introducing $n_2 - n_1$ dummy nodes in[4] $\mathcal{G}_1$. As a strict permutation matrix, $\mathbf{X}$ must also be an orthogonal matrix, *i.e.*, $\mathbf{X}^T\mathbf{X} = \mathbf{X}\mathbf{X}^T = \mathbf{I}_{n_2}$. This fact motivates the following relaxation:

$$J_{vex}(\mathbf{X}) = J_{gm}(\mathbf{X}) - \frac{1}{2}C(\mathbf{X}) = -\frac{1}{2}\sum_i \|\mathbf{A}_i^1\mathbf{X} - \mathbf{X}\mathbf{A}_i^2\|_F^2,$$

$$\text{where } C(\mathbf{X}) = \sum_i \mathrm{tr}(\mathbf{A}_i^1\mathbf{A}_i^1\mathbf{X}\mathbf{X}^T) + \mathrm{tr}(\mathbf{A}_i^2\mathbf{A}_i^2\mathbf{X}^T\mathbf{X}).$$

Observe that due to the orthogonal constraints, $C(\mathbf{X})$ can be considered constant. In addition, maximizing $J_{vex}(\mathbf{X})$ is a convex problem because its Hessian with respect to $\mathrm{vec}(\mathbf{X})$, $-\sum_i(\mathbf{I}\otimes\mathbf{A}_1^i - \mathbf{A}_2^i\otimes\mathbf{I})^T(\mathbf{I}\otimes\mathbf{A}_1^i - \mathbf{A}_2^i\otimes\mathbf{I})$, is always negative semi-definite.

4.2. A concave relaxation

In this section, we introduce a concave relaxation for Eq. 1 assuming $\mathbf{X}$ satisfies the integer constraint.

From Eq. 2, we know that $\mathbf{L}$ is composed by four parts $\mathbf{L} = \mathbf{L}_1 - \mathbf{L}_2 + \mathbf{L}_3 + \mathbf{L}_4$, where

$$\mathbf{L}_1 = \begin{bmatrix} \mathbf{K}_q & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{L}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{K}_q\mathbf{G}_2^T \\ \mathbf{G}_1\mathbf{K}_q & \mathbf{0} \end{bmatrix},$$

$$\mathbf{L}_3 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_1\mathbf{K}_q\mathbf{G}_2^T \end{bmatrix}, \quad \mathbf{L}_4 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_p \end{bmatrix}.$$

Therefore, $J_{gm}(\mathbf{X})$ can be expanded in the following way:

$$\begin{aligned} J_{gm}(\mathbf{X}) = &\ \mathrm{tr}\left(\mathbf{K}_q^T(\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2 \circ \mathbf{G}_1^T\mathbf{X}\mathbf{G}_2)\right) \\ &- \mathrm{tr}\left((\mathbf{K}_q\mathbf{G}_2^T)^T(\mathbf{G}_1^T\mathbf{X} \circ \mathbf{G}_1^T\mathbf{X})\right) \\ &- \mathrm{tr}\left((\mathbf{G}_1\mathbf{K}_q)^T(\mathbf{X}\mathbf{G}_2 \circ \mathbf{X}\mathbf{G}_2)\right) \\ &+ \mathrm{tr}\left((\mathbf{G}_1\mathbf{K}_q\mathbf{G}_2^T)^T(\mathbf{X} \circ \mathbf{X})\right) + \mathrm{tr}\left(\mathbf{K}_p^T(\mathbf{X} \circ \mathbf{X})\right). \end{aligned}$$

The integer constraint in Eq. 1 implies $\mathbf{X}$, $\mathbf{G}_1^T\mathbf{X}$ and $\mathbf{X}\mathbf{G}_2$ are all binary matrices, from which we know that it is equivalent [28, 39] to replace the quadratic terms $\mathbf{X} \circ \mathbf{X}$, $\mathbf{G}_1^T\mathbf{X} \circ \mathbf{G}_1^T\mathbf{X}$ and $\mathbf{X}\mathbf{G}_2 \circ \mathbf{X}\mathbf{G}_2$ by the linear ones $\mathbf{X}$, $\mathbf{G}_1^T\mathbf{X}$ and $\mathbf{X}\mathbf{G}_2$ respectively. This fact leads to the following relaxation:

$$\begin{aligned} J_{cav}(\mathbf{X}) = &\ \mathrm{tr}\left(\mathbf{K}_q^T(\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2 \circ \mathbf{G}_1^T\mathbf{X}\mathbf{G}_2)\right) \\ &- \mathrm{tr}\left((\mathbf{G}_1\mathbf{K}_q\mathbf{G}_2^T)^T\mathbf{X}\right) + \mathrm{tr}\left(\mathbf{K}_p^T\mathbf{X}\right). \end{aligned}$$

Maximizing $J_{cav}(\mathbf{X})$ is a concave problem because its Hessian, $(\mathbf{G}_2 \otimes \mathbf{G}_1)\,\mathrm{diag}(\mathrm{vec}(\mathbf{K}_q))(\mathbf{G}_2 \otimes \mathbf{G}_1)^T$, is positive semi-definite if the edge affinity is positive (*i.e.*, $\mathbf{K}_q \geq \mathbf{0}$).

4.3. A path-following strategy

In this section, we describe a path-following strategy for optimizing Eq. 1. Inspired by [39], we approach the non-convex QP by iteratively optimizing a series of the following sub-problems:

$$\begin{aligned} \max_{\mathbf{X}} \quad & J_\alpha(\mathbf{X}) = (1-\alpha)J_{vex}(\mathbf{X}) + \alpha J_{cav}(\mathbf{X}), \\ \text{s.t.} \quad & \mathbf{X}\mathbf{1}_{n_2} \leq \mathbf{1}_{n_1}, \mathbf{X}^T\mathbf{1}_{n_1} \leq \mathbf{1}_{n_2}, \mathbf{X} \geq \mathbf{0}_{n_1\times n_2}, \end{aligned} \tag{5}$$

where $\alpha \in [0, 1]$ is a tradeoff between the convex relaxation $J_{vex}(\mathbf{X})$ and the concave one $J_{cav}(\mathbf{X})$. When $\alpha = 0$, the problem is a convex optimization problem which has a global optimal solution no matter the choice of the initialization. When $\alpha = 1$, the problem is a concave optimization problem which always leads to an integer solution [5, 28]. The process starts with $\alpha = 0$ and successively increasing α until 1. Fig. 3 illustrates the procedure of optimizing a graph matching problem using this strategy. In Fig. 3a, we demonstrate the objective functions J_α and J_{gm} with respect to the change of α. Note that there is a turning point around $\alpha = 0.12$ in the curve of J_α. This is because at this point the two relaxations achieve the same value, *i.e.*, $J_{vex} = J_{cav}$. As $\alpha \to 1$, the values of J_{gm}, J_α and J_{cav} are getting closer to each other, and meanwhile, $\mathbf{X}$ is turning into a binary matrix (Fig. 3c).

For a specific α, we optimize $J_\alpha(\mathbf{X})$ taking the Frank-Wolfe's algorithm (FW) [17, 25, 39], a simple yet powerful

[4] Without loss of generality, let's assume $n_1 \leq n_2$.

method for nonlinear programming. FW successively update the solution as $\mathbf{X}^* = \mathbf{X}_0 + \lambda\mathbf{Y}$ given an initial $\mathbf{X}_0$. At each step, it needs to compute two components: (1) the optimal direction $\mathbf{Y} \in \mathbb{R}^{n_1 \times n_2}$ and (2) the optimal step size $\lambda \in [0, 1]$. To compute $\mathbf{Y}$, we solve the following linear programming using the Hungarian algorithm:

$$\begin{aligned} \max_{\mathbf{Y}} \quad & \operatorname{tr}\Big(\nabla J_\alpha(\mathbf{X}_0)^T(\mathbf{Y} - \mathbf{X}_0)\Big), \\ \text{s.t.} \quad & \mathbf{Y}\mathbf{1}_{n_2} \leq \mathbf{1}_{n_1}, \mathbf{Y}^T\mathbf{1}_{n_1} \leq \mathbf{1}_{n_2}, \mathbf{Y} \geq \mathbf{0}_{n_1 \times n_2}, \end{aligned}$$

where the gradients can be efficiently computed using matrix operation:

$$\begin{aligned} \nabla J_\alpha(\mathbf{X}) &= (1-\alpha)\nabla J_{vex}(\mathbf{X}) + \alpha \nabla J_{cav}(\mathbf{X}), \\ \nabla J_{vex}(\mathbf{X}) &= 2\mathbf{H}_1(\mathbf{H}_1^T\mathbf{X}\mathbf{H}_2 \circ \mathbf{L})\mathbf{H}_2^T - \mathbf{H}_1(\mathbf{H}_1^T\mathbf{H}_1 \circ \mathbf{U}\mathbf{U}^T)\mathbf{H}_1^T\mathbf{X} \\ &\quad - \mathbf{X}\mathbf{H}_2(\mathbf{H}_2^T\mathbf{H}_2 \circ \mathbf{V}\mathbf{V}^T)\mathbf{H}_2^T, \\ \nabla J_{cav}(\mathbf{X}) &= 2\mathbf{G}_1(\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2 \circ \mathbf{K}_q)\mathbf{G}_2^T - \mathbf{G}_1\mathbf{K}_q\mathbf{G}_2^T + \mathbf{K}_p. \end{aligned}$$

And the line search for the optimal λ can be found in closed form by solving:

$$\max_{\lambda} \quad J_\alpha(\mathbf{X}_0 + \lambda\mathbf{Y}), \quad \text{s.t.} \quad \lambda \in [0, 1].$$

4.4. Other implementation details

A similar path-following strategy was proposed in [39] and its performance over the state-of-the-art methods has been therein demonstrated for solving a less general graph matching problem (*i.e.*, $\operatorname{tr}(\mathbf{A}_1\mathbf{X}\mathbf{A}_2\mathbf{X}^T)$). We performed an extensive study of using this strategy for solving the most general graph matching problem (*i.e.*, $\mathbf{x}^T\mathbf{K}\mathbf{x}$) and we empirically found that it can be improved with the following steps:

Convergence: Although the FW algorithm is easy to implement, it converges sub-linearly. To get faster convergence speed while keeping its advantages in efficiency and low memory cost, we adopt a modified Frank-Wolfe (MFW) [18] to find a better searching direction $\mathbf{Y}$ by a convex combination of previously obtained solutions. As it is shown in Fig. 3b, MFW converges much faster than FW.

The concave-convex structure: $J_\alpha(\mathbf{X})$ is naturally divided into a concave part and a convex one. To take the advantage of this structure, we adopt the concave-convex procedure (CCCP) [38] that approximates a non-convex objective function by a series of linearizations of the concave part given the current solution. In practice, we found that CCCP outperformed an individual FW in the case when $J_\alpha(\mathbf{X})$ is close to a convex one, *i.e.*, α is small. However, the performance of CCCP would downgrade as α gets larger due to the increasing loss in the approximation of the concave part. For instance, Fig. 3b compares CCCP with FW and MFW for optimizing $J_\alpha(\mathbf{X})$. CCCP outperforms MFW when $\alpha = 0.08$. However, MFW converges fastest for $\alpha = 0.30$. Therefore, we adopt CCCP only in the beginning steps when α is smaller than a manually defined threshold η.

Local vs global: Although the path-following strategy returns an integer solution by smoothly tracking the local optima in a convex space, it does not guarantee to obtain the global optimal of the non-convex objective function. An important reason is that at each step, it locally optimizes over $J_\alpha(\mathbf{X})$ instead of the global one $J_{gm}(\mathbf{X})$. And it is possible that $J_\alpha(\mathbf{X})$ gets improved while $J_{gm}(\mathbf{X})$ gets worse. In order to escape from this phenomenon, we keep increasing the global score of $J_{gm}(\mathbf{X})$ during the optimization by discarding the bad temporary solution that worsens the score of $J_{gm}(\mathbf{X})$ and computing an alternative one by applying one step of FW for optimizing $J_{gm}(\mathbf{X})$. This refinement is analogous to the usage of FW in [25]. As shown in Fig. 3a, the performance of the path-following algorithm can be greatly improved by only optimizing over J_{gm} three times.

Algorithm 1: Factorized graph matching

input : $\mathbf{K}_p$, $\mathbf{K}_q$, $\mathbf{G}_1$, $\mathbf{G}_2$, δ, η
output: $\mathbf{X}$

Initialize $\mathbf{X}$ *to be a doubly stochastic matrix*;
Factorize $\mathbf{L} = \mathbf{U}\mathbf{V}^T$ *with SVD*;
for $\alpha = 0 : \delta : 1$ **do** *Path-following*
  if $\alpha \leq \eta$ **then**
    Optimize Eq. 5 via CCCP to obtain $\mathbf{X}^*$;
  else
    Optimize Eq. 5 via MFW to obtain $\mathbf{X}^*$;
  if $J_{gm}(\mathbf{X}^*) < J_{gm}(\mathbf{X})$ **then**
    Optimize Eq. 1 via one step of FW to obtain $\mathbf{X}^*$;
  Update $\mathbf{X} \leftarrow \mathbf{X}^*$;

Algorithm 1 summarizes the workflow of our algorithm. The initial $\mathbf{X}$ can be an arbitrary doubly stochastic matrix. The complexity of our algorithm can be roughly calculated as $O\big(T(\tau_{hun} + \tau_\nabla + \tau_\lambda) + \tau_{\mathbf{L}}\big)$, where T is the number of iterations for the FW and MFW, and $\tau_{\mathbf{L}} = (n_1 + m_1)(n_2 + m_2)^2$ is the cost of computing the SVD of $\mathbf{L}$. The Hungarian algorithm can be finished in $\tau_{hun} = \max(n_1^3, n_2^3)$. The gradient of ∇J_α and the line search of λ incur the same computational cost, $\tau_\nabla = \tau_\lambda = (n_1 + m_1)(n_2 + m_2)$.

5. Experiments

This section reports experimental results on three datasets (one synthetic and two real) and compares our method against seven state-of-the-art algorithms:

Graduated assignment (GA): GA [21] performs gradient ascent on a relaxed Eq. 1 driven by an annealing schedule. At each step, it maximizes a Taylor expansion of the non-convex QP around the previous approximate solution. The accuracy of the approximation is controlled by a continuation parameter, $\beta_{t+1} \leftarrow \alpha\beta_t \leq \beta_{max}$. In all experiments, we set $\alpha = 1.075$, $\beta_0 = .5$ and $\beta_{max} = 200$.

Spectral matching (SM): SM [24] optimizes a relaxed

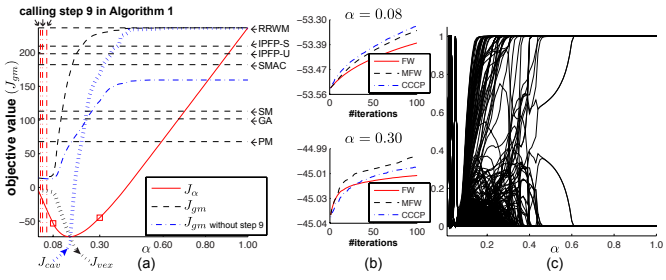

Figure 3. The path-following optimization of the example in Fig. 6b. (a) The comparison of the objectives optimized by our algorithm and other state-of-the-art graph matching methods (See the experiment section for the details of these methods). (b) The comparison of optimizing $J_\alpha(\mathbf{X})$ for different α's. (c) The updating of $\mathbf{X}$, where each curve corresponds a x_{ij}.

problem of Eq. 1 that drops the affine constraints and introduces a unit-length constraint on $\mathbf{x}$, that is:

$$\max_{\mathbf{X}} J_{gm}(\mathbf{X}), \quad \text{s.t. } \mathbf{x}^T\mathbf{x} = 1.$$

The globally optimal solution of the relaxed problem is the leading eigenvector of $\mathbf{K}$.

Spectral matching with affine constraints (SMAC): SMAC [13] adds affine constraints to the SM problem maximizing:

$$\max_{\mathbf{X}} J_{gm}(\mathbf{X}), \quad \text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x}^T\mathbf{x} = 1.$$

The solution is also an eigenvalue problem.

Integer projected fixed point method (IPFP): IPFP [25] is based on FW. It can take any continuous or discrete solution as inputs and iteratively improve the solution. In our experiments, we implemented two versions: (1) IPFP-U, that starts from the same initial $\mathbf{X}$ as our method; (2) IPFP-S, that is initialized by SM.

Probabilistic graph matching (PM): Pm [40] designs the following convex objective function that can be globally optimized by applying the Sinkhorn's algorithm [32]:

$$\min_{\mathbf{X}} D(\mathbf{Y}\|\mathbf{X}), \text{ s.t. } \mathbf{X}\mathbf{1}_{n_2} \le \mathbf{1}_{n_1}, \mathbf{X}^T\mathbf{1}_{n_1} \le \mathbf{1}_{n_2}, \mathbf{X} \ge \mathbf{0}_{n_1 \times n_2},$$

where $D(\mathbf{Y}\|\mathbf{X})$ denotes the relative entropy error and $\mathbf{Y} \in \mathbb{R}^{n_1 \times n_2}$ is calculated by marginalizing $\mathbf{K}$. It is worthwhile pointing out that with our notation, $\mathbf{Y}$ can be computed in a matrix form as $\mathbf{Y} = \mathbf{G}_1\mathbf{K}_q\mathbf{G}_2^T$.

Re-weighted random walk matching (RRWM): RRWM [11] introduces a random walk view on the problem and obtains the solution by simulating random walks with re-weighting jumps enforcing the matching constraints on the association graph. We fixed its parameters $\alpha = 0.2$ and $\beta = 30$ in all experiments.

We used existing code from the author's websites for all methods. Notice that all methods need a post-processing step to discretize $\mathbf{X}$. To make a fair comparison, we applied the Hungarian algorithm to make this discretization in all methods. The parameters for our method were fixed to $\delta = 0.01$ and $\eta = 0.1$ in all experiments. The code was implemented in Matlab on a laptop platform with 2.4G Intel Core 2 Duo and 4G memory. FGM was able to obtain the solution within a minute for graphs with 50 nodes.

We evaluated both the matching accuracy and the objective score for the comparison of performance. The matching accuracy, $\frac{\text{tr}(\mathbf{X}_{alg}^T\mathbf{X}_{tru})}{\text{tr}(\mathbf{1}_{n_2\times n_1}\mathbf{X}_{tru})}$, is calculated by computing the consistent matches between the correspondence matrix $\mathbf{X}_{alg}$ given by algorithm and ground-truth $\mathbf{X}_{tru}$. The objective score, $\frac{J_{gm}(\mathbf{X}_{alg})}{J_{gm}(\mathbf{X}_{ours})}$, is computed as the ratio between the objective values of our method and other algorithms.

5.1. Synthetic dataset

This experiment performed a comparative evaluation of seven algorithms on randomly synthesized graphs following the experimental protocol of [11, 13, 21]. For each trial, we constructed two identical graphs, $\mathcal{G}_1$ and $\mathcal{G}_2$, each of which consists of 20 inlier nodes and later we added n_{out} outlier nodes (in both graphs). For each pair of nodes, the edge is randomly generated according to the edge density parameter $\rho \in [0, 1]$. Each edge in the first graph was assigned a random edge score distributed uniformly as $q_c^1 \sim \mathcal{U}(0, 1)$ and the corresponding edge $q_c^2 = q_c^1 + \epsilon$ in the second graph is perturbed by adding a random Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The edge-affinity matrix $\mathbf{K}_q$ was computed as $k^q_{c_1c_2} = \exp(-\frac{(q^1_{c_1}-q^2_{c_2})^2}{0.15})$ and the node-affinity $\mathbf{K}_p$ was set to zero.

The experiment tested the performance of GM methods under three parameter settings. For each setting, we generated 100 different pairs of graphs and evaluated the average accuracy and objective score. In the first setting (Fig. 4a), we increased the number of outliers from 0 to 20 while fixing the noise $\sigma = 0$ and considering only fully connected graphs (*i.e.*, $\rho = 1$). In the second case (Fig. 4b), we perturbed the edge weights by changing the noise parameter σ from 0 to 0.2, while fixing the other two parameter $n_{out} = 0$ and $\rho = 1$. In the last case (Fig. 4c), we verified the performance of matching sparse graphs by varying ρ from 1 to 0.3. Under varying parameters, it can be observed that in most of cases, our method achieves the best performance over all other algorithms in terms of both accuracy and objective ratio. RRWM is comparable to our method. In particular, it slightly outperforms ours in the case when the graph edges contain large deformation (Fig. 4b). This is because the stochastic scheme adopted by RRWM can update the correspondence matrix more robustly than other optimization-based methods.

5.2. CMU house dataset

The CMU house image sequence [1] is commonly used to test the performance of graph matching algorithms [8, 11, 14, 35]. This dataset consists of 111 frames of a house, each of which has been manually labeled with 30 land-

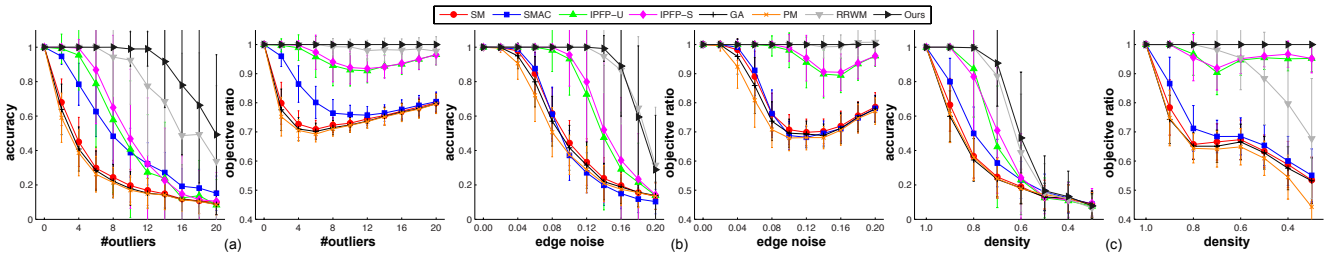

Figure 4. Comparison of graph matching for synthetic datasets. (a) Performance as a function of the outlier number (n_{out}). (b) Performance as a function of the edge noise (σ). (c) Performance as a function of the density of edges (ρ).

marks. We used Delaunay triangulation to connect the landmarks. The edge weight q_c is computed as the pair-wise distance between the connected nodes. Given an image pair, the edge-affinity matrix $\mathbf{K}_q$ was computed by $k^q_{c_1c_2} = \exp(-\frac{(q^1_{c_1}-q^2_{c_2})^2}{2500})$ and the node-affinity $\mathbf{K}_p$ was set to zero. We tested the performance of all methods as a function of the separation between frames. We matched all possible image pairs, spaced exactly by $0:10:90$ frames and computed the average matching accuracy and objective ratio per sequence gap. Fig. 5a demonstrates an example pair of two frames.

We tested the performance of graph matching methods under two scenarios. In the first case (Fig. 5b) we used all 30 nodes (i.e. landmarks) and in the second one (Fig. 5c) we matched sub-graphs by randomly picking 25 landmarks. It can be observed that in the first case (Fig. 5b), RRWM, IPFP-S and our method almost obtained perfect matching of the original graphs. As some nodes became invisible and the graph got corrupted (Fig. 5c), the performance of all the methods degrades. However, our method consistently achieved the best performance.

5.3. Pascal image dataset

The third experiment used the dataset from [26]. This dataset consists of 30 pairs of car images and 20 pairs of motorbike images selected from Pascal 2007 [16]. Each pair contains $30 \sim 60$ ground-truth correspondences. We computed for each node the feature, p_i, as its orientation of the normal vector at that point to the contour where the point was sampled. We adopted the Delaunay triangulation to build graphs and each edge was represented by a couple of values, $\mathbf{q}_c = [d_c, \theta_c]^T$, where d_c is the pair-wise distance between the connected nodes and θ_c is the absolute angle between the edge and the horizontal line. Thus, for each pair of images, we computed the node affinity as $k^p_{ij} = \exp(-|p_i - p_j|)$ and the edge affinity as $k^q_{c_1c_2} = \exp(-\frac{1}{2}|d_{c_1} - d_{c_2}| - \frac{1}{2}|\theta_{c_1} - \theta_{c_2}|)$. Fig. 6a and Fig. 6b demonstrate example pairs of car and motorbike images respectively.

To test the performance against noise, we randomly selected $0 \sim 20$ outlier nodes from the background. In the case when no outliers exist, our method achieves above 80% matching rate in both datasets (Fig. 6bc), which is higher than 75% presented in [26]. From Fig. 6c, it is interesting to see that RRWM performs better in terms of accuracy for particular level of outliers, whereas our method obtains a higher objectives. This is because the ground-truth correspondence may not be always the optimal solution to the problem.

6. Conclusions

This paper presents FGM, a new graph matching algorithm that exploits the properties of the factorized affinity or graph matrix. Three main benefits follow from factorizing the affinity matrix. First, there is no need to explicitly compute the affinity matrix. Second, it provides a unified approach to frame several graph matching algorithms. Third, using the factorization, a new optimization based on FW and CCCP is proposed. Experimental results on synthetic and real datasets illustrate the performance of the new method.

In the paper we have illustrated the advantages of factorizing the pair-wise affinity matrix of typical graph matching problems. The most computationally consuming part of the algorithm is the large number of iterations needed for FW method to converge when J_α is close to a convex function. Therefore, more advanced techniques (*e.g.*, conjugate gradient) can be used to speedup FW. In addition, we are currently exploring the extension of this factorization methods to other higher-order graph matching problems [9, 14, 40] as well as learning parameters for graph matching [8, 26].

Acknowledgements The first author was supported by the National Science Foundation (NSF) under Grant No. EEEC-0540865 and CPS-0931999. The second author was partially supported by the NSF grant RI-1116583. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

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

[2] E. L. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. SIAM, 2003.

[3] H. A. Almohamad and S. O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):522–525, 1993.

[4] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR*, 2005.

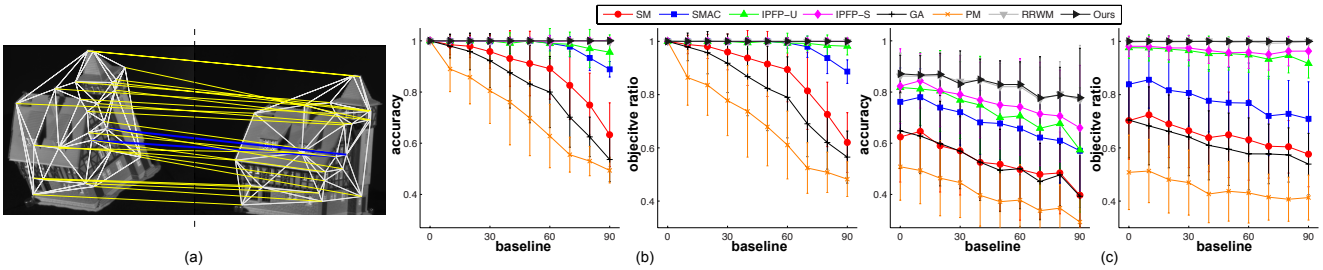

Figure 5. Comparison of graph matching for CMU house datasets. (a) An example pair of frames with the correspondence generated by our method, where the blue lines indicate incorrect matches. (b) Performance when using 30 nodes. (c) Performance when using 25 nodes.

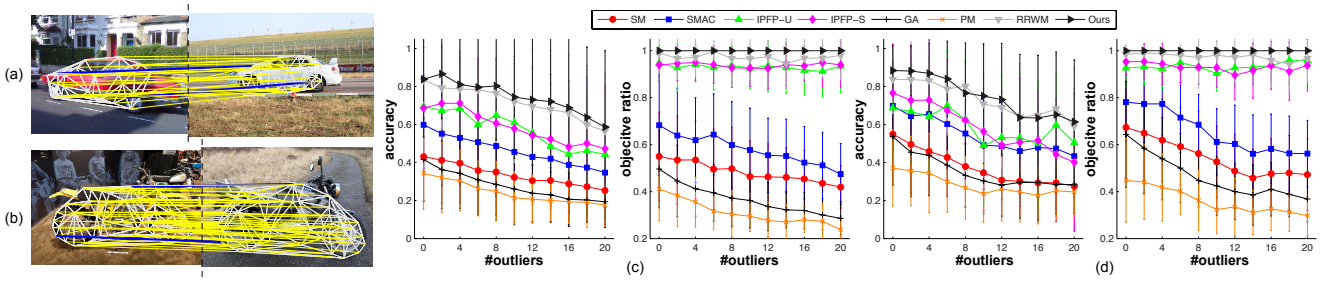

Figure 6. Comparison of graph matching for the Pascal 2007 dataset. (a) An example pair of car images with the correspondence generated by our method, where the blue lines indicate incorrect matches. (b) An example pair of motorbike images. (c) Performance as a function of the outlier number for the car images. (d) Performance as a function of the outlier number for the motorbike images.

[5] I. M. Bomze and G. Danninger. A global optimization algorithm for concave quadratic programming problems. *SIAM J. Optim.*, 3:826–842, 1993.

[6] W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, 2011.

[7] R. Burkard, M. DellAmico, and S. Martello. *Assignment Problems*. SIAM, 2009.

[8] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(6):1048–1058, 2009.

[9] M. Chertok and Y. Keller. Efficient high order matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(12):2205–2215, 2010.

[10] M. Chertok and Y. Keller. Spectral symmetry analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(7):1227–1238, 2010.

[11] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. In *ECCV*, 2010.

[12] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *IJPRAI*, 18(3):265–298, 2004.

[13] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *NIPS*, 2006.

[14] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(12):2383–2395, 2011.

[15] O. Duchenne, A. Joulin, and J. Ponce. A graph-matching kernel for object categorization. In *ICCV*, 2011.

[16] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[17] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[18] M. Fukushima. A modified Frank-Wolfe algorithm for solving the traffic assignment problem. *Transp. Res. Part B: Methodological*, 18(2):169–177, 1984.

[19] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[20] U. Gaur, Y. Zhu, B. Song, and A. Roy-Chowdhury. A "string of feature graphs model" for recognition of complex activities in natural videos. In *ICCV*, 2011.

[21] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(4):377–388, 1996.

[22] J. Hays, M. Leordeanu, A. A. Efros, and Y. Liu. Discovering texture regularity as a higher-order correspondence problem. In *ECCV*, 2006.

[23] H. Jiang, S. X. Yu, and D. R. Martin. Linear scale and rotation invariant matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(7):1339–1355, 2011.

[24] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005.

[25] M. Leordeanu, M. Hebert, and R. Sukthankar. An integer projected fixed point method for graph matching and MAP inference. In *NIPS*, 2009.

[26] M. Leordeanu, R. Sukthankar, and M. Hebert. Unsupervised learning for graph matching. *Int. J. Comput. Vis.*, 95(1):1–18, 2011.

[27] E. M. Loiola, N. M. de Abreu, P. O. Boaventura, P. Hahn, and T. M. Querido. A survey for the quadratic assignment problem. *Eur. J. Oper. Res.*, 176(2):657–690, 2007.

[28] J. Maciel and J. Costeira. A global solution to sparse correspondence problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(2):187–199, 2003.

[29] K. M. Ng. *A continuation approach for solving nonlinear optimization problems with discrete variables*. PhD thesis, Stanford University, 2002.

[30] C. Schellewald, S. Roth, and C. Schnörr. Evaluation of convex optimization techniques for the weighted graph-matching problem in computer vision. In *DAGM-Symposium*, 2001.

[31] C. Schellewald and C. Schnörr. Probabilistic subgraph matching based on convex relaxation. In *EMMCVPR*, 2005.

[32] R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Amer. Math. Soc*, 35(2):876–879, 1964.

[33] J. Sullivan and S. Carlsson. Recognizing and tracking human action. In *ECCV*, 2002.

[34] P. H. S. Torr. Solving Markov random fields using semidefinite programming. In *AISTATS*, 2003.

[35] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *ECCV*, 2008.

[36] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(5):695–703, 1988.

[37] B. J. van Wyk and M. A. van Wyk. A POCS-based graph matching algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(11):1526–1530, 2004.

[38] A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Comput.*, 15(4):915–936, 2003.

[39] M. Zaslavskiy, F. R. Bach, and J.-P. Vert. A path following algorithm for the graph matching problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(12):2227–2242, 2009.

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