Factorized Graph Matching

Feng Zhou and Fernando De la Torre

Abstract—Graph matching (GM) is a fundamental problem in computer science, and it plays a central role to solve correspondence problems in computer vision. GM problems that incorporate pairwise constraints can be formulated as a quadratic assignment problem (QAP). Although widely used, solving the correspondence problem through GM has two main limitations: (1) the QAP is NP-hard and difficult to approximate; (2) GM algorithms do not incorporate geometric constraints between nodes that are natural in computer vision problems. To address aforementioned problems, this paper proposes factorized graph matching (FGM). FGM factorizes the large pairwise affinity matrix into smaller matrices that encode the local structure of each graph and the pairwise affinity between edges. Four are the benefits that follow from this factorization: (1) There is no need to compute the costly (in space and time) pairwise affinity matrix; (2) The factorization allows the use of a path-following optimization algorithm, that leads to improved optimization strategies and matching performance; (3) Given the factorization, it becomes straight-forward to incorporate geometric transformations (rigid and non-rigid) to the GM problem. (4) Using a matrix formulation for the GM problem and the factorization, it is easy to reveal commonalities and differences between different GM methods. The factorization also provides a clean connection with other matching algorithms such as iterative closest point; Experimental results on synthetic and real databases illustrate how FGM outperforms state-of-the-art algorithms for GM. The code is available at `http://humansensing.cs.cmu.edu/fgm`.

Index Terms—Graph matching, Feature matching, Quadratic assignment problem, Iterative closet point method.

1 Introduction

Establishing correspondence between two sets of visual features is the key of many computer vision tasks, such as object tracking [1], structure-from-motion [2], and image classification [3]. While solving the correspondence between images is still an open problem in computer vision, much progress has been done in the last decades. Early works such as RANSAC [4] and iterative closest point (ICP) [5] assume that the location of image features are constrained explicitly (*e.g.*, a planar affine transformation) or implicitly (*e.g.*, epipolar ones) by a parametric form. While these methods achieved great success in exploring consistent correspondence between feature points, they are limited to correspondence problems with rigid transformations. Unlike conventional methods, graph matching (GM) formulates the correspondence problem as solving the matching between two graphs. In the past decades, GM has found wide application to address several problems such as shape matching in 2-D [6] and 3-D [7], object categorization [8], [9], feature tracking [1], symmetry analysis [10], action recognition [11], [12], kernelized sorting [13], and protein alignment [14]. Compared to RANSAC and ICP, GM incorporates pairwise node interactions which are important features when matching structural objects (*e.g.*, human bodies). Fig. 1 illustrates an example of matching two graphs, where the nodes are image locations returned by a body part detector. Matching the graphs using only similarity between nodes (*i.e.*, features) might lead to undesirable results because some features of nodes (*e.g.*, hands, feet) are likely to be similar. GM adds pairwise information between nodes (*e.g.*, length of the limbs, orientation of edges) that constraints the problem and better correspondences are found.

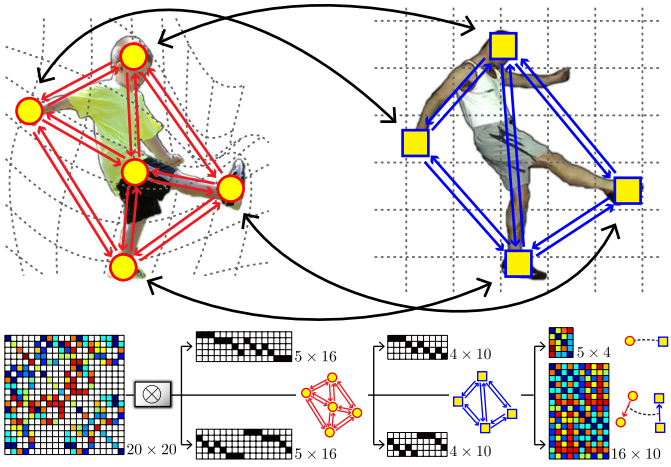

Fig. 1. Matching two human bodies with 5 and 4 features using FGM. FGM simultaneously estimates the correspondence and a smooth non-rigid transformation between shapes. FGM is able to factorize the 20×20 pairwise affinity matrix as a Kronecker product of six smaller matrices. The first two groups of matrices of size 5×16 and 4×10 encode the structure of each of the graphs. The last two matrices encode the affinities for nodes (5×4) and edges (16×10).

Although extensive research has been done for decades, there are still two main challenges in solving GM. (1) Mathematically, GM is formulated as a quadratic assignment problem (QAP) [15]. Unlike the linear assignment problem, which can be efficiently solved with the Hungarian algorithm [16], the QAP is known to be NP-

F. Zhou and F. De la Torre are with Robotics Institute, Carnegie Mellon University.

hard. Therefore, the main body of research in GM has focused on devising more accurate algorithms to solve it approximately. Nevertheless, approximating GM by relaxing the combinatorial constraints is still a challenging problem. This is mainly because the objective function is in general non-convex and thus existing methods are prone to local optima. (2) Many matching problems in computer vision naturally require global constraints among nodes. For instance, given two sets of coplanar points in two images, the matching between points under orthographic projection should be constrained by an affine transformation. Similarly, when matching the deformations of non-rigid objects between two consecutive images, that deformation is typically smooth in space and time. Existing GM algorithms do not constrain the nodes of both graphs to a given geometric transformation (*e.g.*, similarity, affine or non-rigid). For instance, how to constrain the global deformation of the human body configuration shown in Fig. 1.

In order to address these issues, this paper presents factorized graph matching (FGM), a novel framework for optimizing and constraining GM problems. The key idea behind FGM is a closed-form factorization of the pairwise affinity matrix that decouples the local graph structure of the nodes and edge similarities. This factorization is general and can be applied to both undirected and directed graphs. For instance, consider the matching of the two human bodies shown in Fig. 1. The body configurations are represented by two directed graphs with 5 and 4 features, and 16 and 10 edges respectively. Conventional GM algorithms need to construct a large pairwise affinity matrix (20-by-20 in this case). Whereas FGM only requires six small matrices to be computed. In our example (Fig. 1), the first two binary matrices are of dimension 5-by-16 and the second two of dimensions 4-by-10 and describe the local structure of the first and second graph. The last two matrices are of dimensions 5-by-4 and 16-by-10, and they encode the similarity between nodes and edges respectively.

The main contribution of this paper is to propose the factorization of the affinity matrix and illustrate four main benefits of this factorization in GM problems:

- Using the factorization, there is no need to compute the expensive (in space and time) affinity matrix, which computational cost scales quartically with the number of features.
- We derive a new path-following algorithm that leads to improved optimization strategies and matching performance. This is possible because using the factorization it is relatively easy to provide a concave and convex approximation, without the proposed factorization it is unclear how to derive this path-following algorithm.
- A major contribution of this work is to incorporate global geometric constraints into GM. To the best of our knowledge, this is the first work that addresses adding non-rigid constraints into GM. This is possible due to the factorization, which decouples the local structure in each of the graphs.
- We provide a simple and compact matrix notation to formulate GM problems. Using the factorization and the matrix formulation, it is very easy to understand the commonalities and differences between GM methods and relate them to other problems like ICP and Markov Random Fields (MRF).

2 Previous work on GM

In this section, we review previous work on GM with a unified matrix formulation. Formulating GM in matrix form has several benefits beyond simplicity of understanding that will be discussed through the paper. Section A introduces the GM problem and formulates GM as a quadratic assignment problem in matrix form. Section B discusses several advances of GM methods.

2.1 Definition of GM

To better understand the difference among previous work, we provided here a brief overview of the graph matching problem and its mathematical definition. We denote (see notation[1]) a graph with n nodes and m directed edges as a 4-tuple $\mathcal{G} = \{\mathbf{P}, \mathbf{Q}, \mathbf{G}, \mathbf{H}\}$. The features for nodes and edges are specified by $\mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_n] \in \mathbb{R}^{d_p \times n}$ and $\mathbf{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_m] \in \mathbb{R}^{d_q \times m}$ respectively, where d_p and d_q are the dimensionality of the features. For instance, $\mathbf{p}_i$ could be a 128-D SIFT histogram extracted from the image patch around the i^{th} node and $\mathbf{q}_c$ could be a 2-D vector that encodes the length and orientation of the c^{th} edge. The topology of the graph is encoded by two node-edge incidence matrices $\mathbf{G}, \mathbf{H} \in \{0,1\}^{n \times m}$, where $g_{ic} = h_{jc} = 1$ if the c^{th} edge starts from the i^{th} node and ends at the j^{th} node. Fig. 2a illustrates two synthetic graphs, whose edge connection between nodes is encoded by the binary matrices shown in Fig. 2b-c. In our previous work [17], a simpler representation of graph was adopted and valid only for undirected graphs. This representation is more general and applicable for both undirected and directed graphs. Directed graphs occur when the edge features are asymmetrical such as the angle between an edge and the horizontal line.

Given a pair of graphs, $\mathcal{G}_1 = \{\mathbf{P}_1, \mathbf{Q}_1, \mathbf{G}_1, \mathbf{H}_1\}$ and $\mathcal{G}_2 = \{\mathbf{P}_2, \mathbf{Q}_2, \mathbf{G}_2, \mathbf{H}_2\}$, we compute two affinity matrices, $\mathbf{K}_p \in \mathbb{R}^{n_1 \times n_2}$ and $\mathbf{K}_q \in \mathbb{R}^{m_1 \times m_2}$, to measure 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^{th}_1 node of $\mathcal{G}_1$ and the i^{th}_2 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

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} or $[\mathbf{X}]_{ij}$ denotes 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. $|\mathbf{X}|$ represents the determinant of the square matrix $\mathbf{X}$. $\mathrm{vec}(\mathbf{X})$ 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.

c_1^{th} edge of $\mathcal{G}_1$ and the c_2^{th} edge of $\mathcal{G}_2$. For instance, Fig. 2d illustrates an example pair of $\mathbf{K}_p$ and $\mathbf{K}_q$ for the two synthetic graphs.

Given two graphs and the associated affinity matrices, the problem of GM consists in finding the optimal correspondence $\mathbf{X}$ between nodes, such that the sum of the node and edge compatibility is maximized:

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

where $\mathbf{X} \in \Pi$ is constrained to be a one-to-one mapping, *i.e.*, Π is the set of partial permutation matrices:

$$\Pi = \{\mathbf{X} | \mathbf{X} \in \{0,1\}^{n_1 \times n_2}, \mathbf{X}\mathbf{1}_{n_2} \leq \mathbf{1}_{n_1}, \mathbf{X}^T \mathbf{1}_{n_1} = \mathbf{1}_{n_2}\}. \quad (2)$$

The inequality in the above definition is used for the case when the graphs are of different sizes. Without loss of generality, we assume $n_1 \geq n_2$ from now on. For instance, the matrix $\mathbf{X}$ shown in the top row of Fig. 2e defines the node correspondence shown in Fig. 2a.

To be concise in notation, we encode the node and edge affinities in a symmetrical matrix $\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$, whose elements are computed as follows:

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

where the diagonal and off-diagonal elements encode the similarity between nodes and edges respectively.

Using $\mathbf{K}$, GM can be concisely formulated as the following quadratic assignment problem (QAP) [15]:

$$\max_{\mathbf{X} \in \Pi} \quad J_{gm}(\mathbf{X}) = \mathrm{vec}(\mathbf{X})^T \mathbf{K}\, \mathrm{vec}(\mathbf{X}). \quad (3)$$

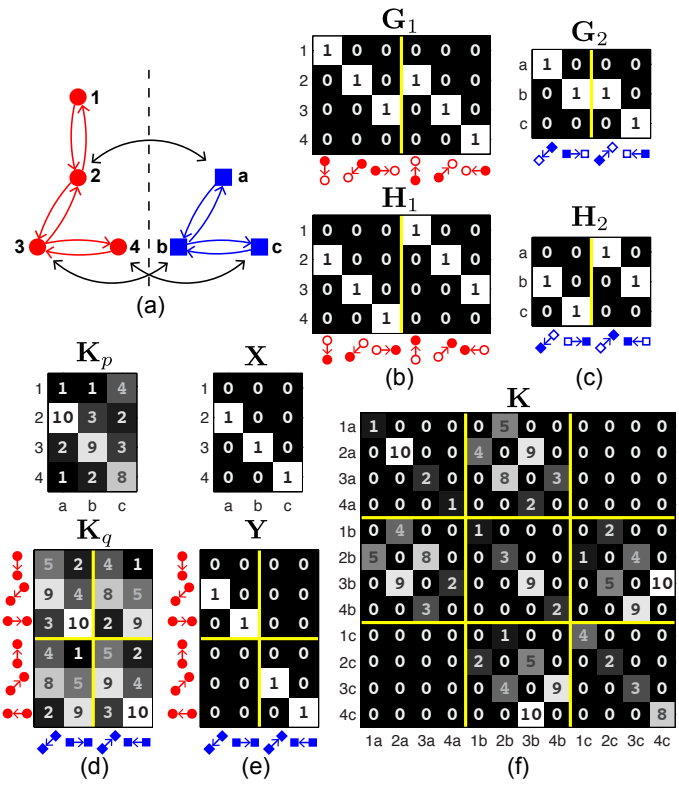

Fig. 2. An example GM problem. (a) Two synthetic graphs. (b) The 1^{st} graph's incidence matrices $\mathbf{G}_1$ and $\mathbf{H}_1$, where the non-zero elements in each column of $\mathbf{G}_1$ and $\mathbf{H}_1$ indicate the starting and ending nodes in the corresponding directed edge, respectively. (c) The 2^{nd} graph's incidence matrices $\mathbf{G}_2$ and $\mathbf{H}_2$. (d) The node affinity matrix $\mathbf{K}_p$ and the edge affinity matrix $\mathbf{K}_q$. (e) The node correspondence matrix $\mathbf{X}$ and the edge correspondence matrix $\mathbf{Y}$. (f) The global affinity matrix $\mathbf{K}$.

2.2 Advances on GM

As a generic problem for matching structural data, GM has been studied for decades in computer science and mathematics [18]. Early works [19]–[21] on GM often treated the problem as a special case of (sub)graph isomorphism, where the graphs are binary and an exact matching between nodes and edges is of interest. Nevertheless, the stringent constraints imposed by exact matching are too rigid for real applications in computer vision. Modern works focus on finding an inexact matching between graphs with weighted attributes on nodes and edges. Due to the combinatorial nature, however, globally optimizing GM is NP-hard. A relaxation of the permutation constraints (Eq. 2) is necessary to find an approximation to the problem. Based on the approximation strategy, previous work (see Fig. 3 for a summary) can be broadly categorized in three groups: spectral relaxation, semidefinite-programming (SDP) relaxation and doubly-stochastic relaxation.

The first group of methods approximates the permutation matrix with an orthogonal one, *i.e.*, $\mathbf{X}^T\mathbf{X} = \mathbf{I}$. Under the orthogonal constraint, optimizing GM can be solved in closed-form as an eigen-value problem [22]–[25]. However, these methods can only work for the Koopmans-Beckmann's QAP [26], a special case of QAP (See Section 5 for more details). To handle more complex problems in computer vision, Leordeanu and Hebert [27] approximated Eq. 3 by relaxing $\mathbf{X}$ to be of unit length, *i.e.*, $\|\mathrm{vec}(\mathbf{X})\|_2 = 1$. The optimal $\mathbf{X}$ can then be efficiently computed as the leading eigen-vector of $\mathbf{K}$. Cour *et al.* [28] incorporated an affine constraint to solve a more general spectral problem, hence finding better approximations to the original problem.

As a general tool for approximating combinatorial problems, SDP was also used to approximate GM. In [29], [30], the authors reformulated the objective of GM by introducing a new variable $\mathbf{Y} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ subject to $\mathbf{Y} = \mathrm{vec}(\mathbf{X})\,\mathrm{vec}(\mathbf{X})^T$. SDP approximates GM by relaxing the non-convex constraint on $\mathbf{Y}$ as a convex semi-definite one, $\mathbf{Y} - \mathrm{vec}(\mathbf{X})\,\mathrm{vec}(\mathbf{X})^T \succeq 0$. After computing $\mathbf{Y}$ using SDP, the correspondence $\mathbf{X}$ can be approximated by a randomized algorithm [29] or a winner-take-all strategy [30]. The main advantage of using SDP is its theoretical guarantees [31] to find a polynomial time 0.879 approximation to many NP-hard problems. However, in practice it is too expensive to use SDP because the

variable $\mathbf{Y}$ squares the problem size.

Most methods relax $\mathbf{X} \in \mathcal{D}$ to be a doubly stochastic matrix, the convex hull of the permutation matrices:

$$\mathcal{D} = \{\mathbf{X} | \mathbf{X} \in [0,1]^{n_1 \times n_2}, \mathbf{X}\mathbf{1}_{n_2} \leq \mathbf{1}_{n_1}, \mathbf{X}^T\mathbf{1}_{n_1} = \mathbf{1}_{n_2}\}. \quad (4)$$

Under this constraint, optimizing GM can be treated as a non-convex quadratic programming problem and various strategies have been proposed to find a local optima. Almohamad and Duffuaa [32] approximated the quadratic cost using linear programming. Zaslavskiy *et al*. [33] employed a path-following strategy to approach the non-convex quadratic programming problem. Recently, Liu *et al*. [34] extended the path-following algorithm for matching asymmetrical adjacency matrices in more general problems. However, these works are only applicable to Koopmans-Beckmann's QAP and it is unclear how to apply it to the more general Lawler's QAP problem [35]. Gold and Rangarajan [36] proposed the graduated assignment algorithm to iteratively solve a series of linear approximations of the cost function using Taylor expansions. Its convergence has been recently studied and improved in [37] with a soft constrained mechanism. Van Wyk and Van Wyk [38] proposed to iteratively project the approximate correspondence matrix onto the convex domain of the desired integer constraints. Leordeanu *et al*. [39] proposed an integer projection algorithm to optimize the objective function in an integer domain. Torresani *et al*. [40] designed a complex objective function which can be efficiently optimized by dual decomposition. In addition to the optimization-based work, probabilistic frameworks were shown to be useful for interpreting and solving GM problems. Egozi *et al*. [41] presented a probabilistic interpretation of the spectral matching algorithm [27], which is a maximum-likelihood estimate of the assignment probabilities. Zass and Shashua [42] proposed a convex relative-entropy error that emerges from a probabilistic interpretation of the GM problem. Inspired by the PageRank algorithm, Cho *et al*. [43] introduced a random-walk algorithm to approximate GM problem.

In addition to the efforts on devising better approximations for Eq. 3, there are three advances in GM methods leading to improved results: learning the pairwise affinity matrix [44], [45], incorporate high-order features [42], [46], and progressive mechanism [47]. First, it has been shown that a better $\mathbf{K}$ for GM can be learned in an unsupervised [44] or a supervised [45] manner. Second, the matrix $\mathbf{K}$ encoding the pairwise geometry is susceptible to scale and rotation differences between sets of points. To make GM invariant to rigid deformations, [42], [46] extended the pairwise $\mathbf{K}$ to a tensor that encodes high-order geometrical relations. However, a small increment in the order of relations leads to a combinatorial explosion of the amount of data needed to support the algorithm. Therefore, most of high-order GM methods can only work on very sparse graphs with no more than 3-order features. In addition, it is unclear on how to extend high-order methods to incorporate non-rigid deformations. Third, the performance of GM methods in real applications is often limited by the initial construction of graphs. To resolve this issue, Cho and Lee [47] proposed a progressive framework which combines probabilistic progression of graphs with matching of graphs. The algorithm re-estimates in a Bayesian manner the most plausible target graphs based on the matching result, and guarantees to boost the matching objective at subsequent steps.

	$\mathrm{tr}(\mathbf{X}^T\mathbf{A}_1\mathbf{X}\mathbf{A}_2) + \mathrm{tr}(\mathbf{K}_p^T\mathbf{X})$ (Koopmans-Beckmann's QAP)	$\mathrm{vec}(\mathbf{X})^T\mathbf{K}\,\mathrm{vec}(\mathbf{X})$ (Lawler's QAP)
Spectral Relaxation	Umeyama [22] Scott & Longuet-Higgins [23] Shapiro & Brady [24] Caelli & Kosinov [25]	Leordeanu & Hebert [27] Cour *et al.* [28]
SDP Relaxation		Torr [29] Schellewald & Schnörr [30]
Doubly-stochastic Relaxation	Almohamad & Duffuaa [32] Zaslavskiy *et al.* [33] Liu *et al.* [34]	Gold & Rangarajan [36] Tian *et al.* [37] Van Wyk & Van Wyk [38] Leordeanu *et al.* [39] Torresani *et al.* [40] Egozi *et al.* [41] Zass & Shashua [42] Cho *et al.* [43] Ours

Fig. 3. Different relaxations on the constraints for GM.

Unlike most of previous work, we tackle the GM problems from a different perspective. We first derived a principled factorization of the affinity matrix $\mathbf{K}$. This factorization enables the utilization of a path-following algorithm that leads to state-of-the-art performance in approximating GM problems. Thanks to the factorization, we could further introduce global geometrical constraints to better handle non-rigid deformation in GM.

3 Previous Work on ICP

This section describes the ICP method, that as we will see in future sections has a close connection to GM.

3.1 Definition of ICP

Given two sets of points, $\mathbf{P}_1 = [\mathbf{p}_1^1, \cdots, \mathbf{p}_{n_1}^1] \in \mathbb{R}^{d \times n_1}$ and $\mathbf{P}_2 = [\mathbf{p}_1^2, \cdots, \mathbf{p}_{n_2}^2] \in \mathbb{R}^{d \times n_2}$, the iterative closest point (ICP) algorithm (*e.g.*, [5]) aims to find the correspondence and the geometric transformation between points such that the sum of distances is minimized:

$$\min_{\mathbf{X} \in \Pi, \mathcal{T} \in \Psi} \; J_{icp}(\mathbf{X}, \mathcal{T}) = \sum_{i_1 i_2} x_{i_1 i_2} \|\mathbf{p}_{i_1}^1 - \tau(\mathbf{p}_{i_2}^2)\|_2^2 + \psi(\mathcal{T}), \quad (5)$$

where $\mathbf{X} \in \{0,1\}^{n_1 \times n_2}$ denotes the correspondence between points. Depending on the problem, $\mathbf{X}$ denotes either a one-to-one or many-to-one matching. In this paper, we consider a one-to-one matching between points and $\mathbf{X}$ is thus constrained to be a permutation matrix *i.e.*, $\mathbf{X} \in \Pi$, where Π is defined as Eq. 2. $\tau(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ denotes a geometric transformation parameterized by $\mathcal{T}$. The transformation is softly penalized by the function

	Similarity	Affine	RBF Non-rigid
$\mathcal{T}$	$s \in \mathbb{R}$ $\mathbf{R} \in \mathbb{R}^{2\times 2}$ $\mathbf{t} \in \mathbb{R}^2$	$\mathbf{V} \in \mathbb{R}^{2\times 2}$ $\mathbf{t} \in \mathbb{R}^d$	$\mathbf{W} \in \mathbb{R}^{2\times n}$
$\tau(\mathbf{p})$	$s\mathbf{R}\mathbf{p}+\mathbf{t}$	$\mathbf{V}\mathbf{p}+\mathbf{t}$	$\mathbf{p}+\mathbf{W}\phi(\mathbf{p})$
$\tau(\mathbf{P})$	$s\mathbf{R}\mathbf{P}+\mathbf{t}\mathbf{1}_n^T$	$\mathbf{V}\mathbf{P}+\mathbf{t}\mathbf{1}_n^T$	$\mathbf{P}+\mathbf{W}\mathbf{L}_p$
$\psi(\mathcal{T})$	0	0	$\lambda_w \operatorname{tr}(\mathbf{W}\mathbf{L}_p\mathbf{W}^T)$
Ψ	$\mathbf{R}^T\mathbf{R}=\mathbf{I}_2$ $\lvert\mathbf{R}\rvert=1$	$\emptyset$	$\emptyset$

Fig. 4. Parameterization of three geometrical transformations in 2-D space, where $\mathcal{T}$, $\tau(\cdot)$, Ψ and $\psi(\cdot)$ denote the parameter set, transformation function, constraint set and penalization function respectively.

$\psi(\cdot)$ and constrained within the set Ψ. Fig. 4 lists the parameterization of three common transformations: similarity, affine and RBF non-rigid transformation.

Similarity transformation: Given a point $\mathbf{p} \in \mathbb{R}^2$ or a point set $\mathbf{P} \in \mathbb{R}^{2\times n}$, its similarity transformation is computed as $\tau(\mathbf{p}) = s\mathbf{R}\mathbf{p}+\mathbf{t}$ or $\tau(\mathbf{P}) = s\mathbf{R}\mathbf{P}+\mathbf{t}\mathbf{1}_n^T$, where $s \in \mathbb{R}$, $\mathbf{R} \in \mathbb{R}^{2\times 2}$ and $\mathbf{t} \in \mathbb{R}^2$ denote the scaling factor, rotation matrix and translation vector respectively. In addition, the rotation matrix $\mathbf{R} \in \Psi$ has to satisfy the constraints $\Psi = \{\mathbf{R} | \mathbf{R}^T\mathbf{R} = \mathbf{I}_2, |\mathbf{R}| = 1\}$.

Affine transformation: Given a point $\mathbf{p} \in \mathbb{R}^2$ or a point set $\mathbf{P} \in \mathbb{R}^{2\times n}$, its affine transformation is computed as $\tau(\mathbf{p}) = \mathbf{V}\mathbf{p}+\mathbf{t}$ or $\tau(\mathbf{P}) = \mathbf{V}\mathbf{P}+\mathbf{t}\mathbf{1}_n^T$, where $\mathbf{V} \in \mathbb{R}^{2\times 2}$ and $\mathbf{t} \in \mathbb{R}^2$ denote the affine matrix and translation vector respectively.

RBF non-rigid transformation: The parameterization of the RBF non-rigid transformation [48] depends on the choice of the basis points. We choose the basis points as the nodes of the second graph. Given a point $\mathbf{p} \in \mathbb{R}^2$ or a point set $\mathbf{P} \in \mathbb{R}^{2\times n}$, the transformation is computed as a displacement shifted from its initial position, *i.e.*, $\tau(\mathbf{p}) = \mathbf{p}+\mathbf{W}\phi(\mathbf{p})$ or $\tau(\mathbf{P}) = \mathbf{P}+\mathbf{W}\mathbf{L}_p$, where $\mathbf{W} \in \mathbb{R}^{2\times n_2}$ is the weight matrix and $\phi(\mathbf{p}) = [\phi_1(\mathbf{p}), \cdots, \phi_{n_2}(\mathbf{p})]^T \in \mathbb{R}^{n_2}$ denotes the n_2 displacement functions corresponding to the basis points. Each displacement function, $\phi_i(\mathbf{p}) = \exp(-\|\mathbf{p}-\mathbf{p}_i^2\|_2^2/\sigma_w^2)$, is computed between $\mathbf{p}$ and the basis $\mathbf{p}_i^2$ with the bandwidth σ_w. $\mathbf{L}_p = [\phi(\mathbf{p}_1^2), \cdots, \phi(\mathbf{p}_{n_2}^2)] \in \mathbb{R}^{n_2\times n_2}$ is the RBF kernel defined on all basis pairs. Following [48], we regularize the non-smoothness of the displacement field, *i.e.*, $\psi(\mathcal{T}) = \operatorname{tr}(\mathbf{W}\mathbf{L}_p\mathbf{W}^T)$.

3.2 Advances on ICP

In the past decade, ICP has been widely used to solve correspondence problems in image, shape, and surface registration [49] due to its simplicity and low computational complexity. ICP methods can be classified by the type of deformation they recover as rigid or non-rigid.

A major limitation of ICP methods is that they are highly sensitive to the initialization. There have been various strategies proposed to address this issue [50]. One common choice is to introduce structural information into the registration schemes [7], [51]. For instance, Belongie *et al*. [52] proposed shape context, a robust shape representation for finding correspondence between deformed shapes. Advanced optimization scheme can also be employed to improve the performance of ICP. For instance, Chui and Rangarajan [53] proposed to combine soft-assignment with deterministic annealing for non-rigid point registration. This work has been recently extended in [48] for both rigid and non-rigid registration. The most closely related work to our method is the statistical approach [54], [55], where a binary neighborhood graph is used for guiding the ICP minimization for finding better correspondence. Unlike existing ICP algorithms, FGM introduces pairwise constraints that makes the matching problem less prone to local minima.

Alternatively, RANSAC [4] type of algorithms have been widely used to find reliable correspondences between two or more images in the presence of outliers. RANSAC estimates a global relation that fits the data, while simultaneously classifying the data into inliers and outliers. Unlike RANSAC-type of algorithms, FGM is able to efficiently model non-rigid transformations and large rigid transformations.

4 Factorized Graph Matching

This section proposes a novel factorization of the pairwise affinity matrix $\mathbf{K}$. As we will see in the following sections, this factorization provides a light-weight representation for GM problems, allows a unification of GM methods, elaborates a better optimization strategy and makes it easy to add geometric constraints to GM.

The pairwise affinity matrix $\mathbf{K}$ plays a central role in GM because it encodes all the first-order and second-order relations between graphs. Two properties of $\mathbf{K}$, full-rankness and indefiniteness, pose key challenges to conventional GM methods such as spectral methods [27], [28] and gradient-based ones [38], [39], [42], [43]. Fig. 5 illustrates an empirical study on a thousand $\mathbf{K}$s computed from three realistic benchmark datasets in Section 8. The first row of Fig. 5 shows that the ratio between the rank and the dimension of $\mathbf{K}$ equaled to one, *i.e.*, $\mathbf{K}$ was a full-rank matrix in all the experiments. This fact explains why the spectral methods [27], [28] usually perform worse than others: spectral methods employ the rank-one approximation of $\mathbf{K}$ which inevitably leads to loss in accuracy. The second row of Fig. 5 shows that $\mathbf{K}$ was an indefinite matrix because the ratio between its maximum and minimum eigenvalues was smaller than -0.4 in the experiments. The indefiniteness of $\mathbf{K}$ indicates that the maximization of $\operatorname{vec}(\mathbf{X})^T\mathbf{K}\operatorname{vec}(\mathbf{X})$ is a non-convex problem and thus the gradient-based methods [38], [39], [42], [43] is prone to local optima. Instead of treating $\mathbf{K}$ as a black box, we propose a factorization that decouples the structure of $\mathbf{K}$. To the best of our knowledge, this work is the first to propose a closed-form factorization for $\mathbf{K}$ and its applications to GM problems.

To illustrate the intuition behind the factorization, let us revisit the synthetic problem shown in Fig. 2. Notice that $\mathbf{K} \in \mathbb{R}^{n_1n_2\times n_1n_2}$ (Fig. 2f) is composed by two types of affinities: the node affinity ($\mathbf{K}_p$) on its diagonal and the

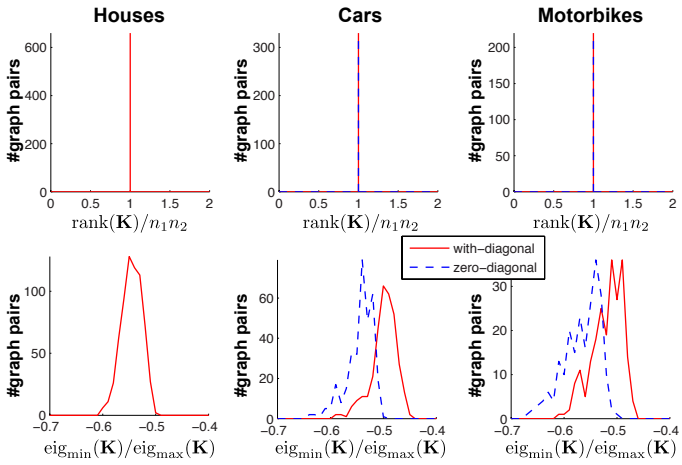

Fig. 5. Statistics of $\mathbf{K}$ computed for the graphs extracted from three real image datasets used in the experiments. The histograms in the top row illustrate the ratio between the rank and the dimension of $\mathbf{K}$. The histograms in the bottom row show the ratio between the minimum and the maximum eigenvalues of the $\mathbf{K}$. For the last two datasets (columns), we test $\mathbf{K}$ with zero values on the diagonal as well. The top and bottom rows indicate that all the computed $\mathbf{K}$ are full-rank and indefinite respectively.

pairwise edge affinity ($\mathbf{K}_q$) on its off-diagonals. Without the diagonal, $\mathbf{K}$ is a sparse block matrix with three unique structures: (1) $\mathbf{K}$ is composed by n_2-by-n_2 smaller blocks $\mathbf{K}_{ij} \in \mathbb{R}^{n_1 \times n_1}$. (2) Some of the $\mathbf{K}_{ij}$s are empty if there is no edge connecting the i^{th} and j^{th} nodes of $\mathcal{G}_2$. These empty blocks can be indexed by $\mathbf{G}_2\mathbf{H}_2^T$, *i.e.*, $\mathbf{K}_{ij} = \mathbf{0}$ if $[\mathbf{G}_2\mathbf{H}_2^T]_{ij} = 0$. (3) For the non-empty blocks, $\mathbf{K}_{ij}$ can be computed in a closed form as $\mathbf{G}_1 \operatorname{diag}(\mathbf{k}_c^q)\mathbf{H}_1^T$, where c is the index of the edge connecting the i^{th} and j^{th} nodes of $\mathcal{G}_2$, *i.e.*, $g_{ic}^2 = h_{jc}^2 = 1$. Based on these observations, and after some linear algebra, it can be shown that $\mathbf{K}$ can be factorized exactly as:

$$\mathbf{K} = \operatorname{diag}(\operatorname{vec}(\mathbf{K}_p)) + (\mathbf{G}_2 \otimes \mathbf{G}_1) \operatorname{diag}(\operatorname{vec}(\mathbf{K}_q))(\mathbf{H}_2 \otimes \mathbf{H}_1)^T. \tag{6}$$

This factorization decouples the graph structure ($\mathbf{G}_1$, $\mathbf{H}_1$, $\mathbf{G}_2$ and $\mathbf{H}_2$) from the similarity ($\mathbf{K}_p$ and $\mathbf{K}_q$).

Eq. 6 is the key contribution of this work. Previous work on GM explicitly computed the computationally expensive (in space and time) $\mathbf{K}$, which is of size $O(n_1^2 n_2^2)$. On the contrary, Eq. 6 offers an alternative framework by replacing $\mathbf{K}$ with six smaller matrices, which are of size $O(n_1 m_1 + n_2 m_2 + n_1 n_2 + m_1 m_2)$. For instance, plugging Eq. 6 into Eq. 3 leads to an equivalent objective function:

$$J_{gm}(\mathbf{X}) = \operatorname{tr}\left(\mathbf{K}_p^T \mathbf{X}\right) + \operatorname{tr}\left(\mathbf{K}_q^T (\underbrace{\mathbf{G}_1^T \mathbf{X} \mathbf{G}_2}_{\mathbf{Y}} \circ \mathbf{H}_1^T \mathbf{X} \mathbf{H}_2)\right), \tag{7}$$

where $\mathbf{Y} \in \{0,1\}^{m_1 \times m_2}$ can be interpreted as a correspondence matrix for edges, *i.e.*, $y_{c_1 c_2} = 1$ if both nodes of c_1^{th} edge in $\mathcal{G}_1$ are matched to the nodes of c_2^{th} edge in $\mathcal{G}_2$. For instance, Fig. 2d illustrates the node and edge correspondence matrices for the matching defined in Fig. 2a.

5 A UNIFIED FRAMEWORK FOR GM

In past decades, a myriad of GM methods have been proposed for solving a variety of applications. Previous GM methods varied in their objective formulations and optimization strategies. Although much effort [18], [56]–[58] has been made on comparing GM methods, a mathematical framework to connect the various GM methods in a coherent manner is lacking. This section derives a unified framework to cast many GM methods using our proposed factorization. Beyond the unification of GM methods, we show that a close relation exists between GM and ICP objectives. This insight allows us to augment GM problems with additional geometrical constraints that are necessary in many computer vision applications. Moreover, we extend ICP to incorporate pair-wise constraints.

5.1 Unification of GM methods

Eq. 3 summarizes one of the most important GM problems studied in recent research on computer vision [27]–[30], [36]–[43]. However, the way of formulating GM is not unique. In other applications [20]–[22], [25], [32], [33], the goal of GM was alternatively formulated as:

$$\max_{\mathbf{X} \in \Pi} \quad \operatorname{tr}(\mathbf{K}_p^T \mathbf{X}) + \operatorname{tr}(\mathbf{A}_1 \mathbf{X} \mathbf{A}_2 \mathbf{X}^T), \tag{8}$$

where the main difference from Eq. 3 is in the second term, which measures the edge compatibility as the linear similarity between two weighted adjacency matrices $\mathbf{A}_1 \in [0,1]^{n_1 \times n_1}$ and $\mathbf{A}_2 \in [0,1]^{n_2 \times n_2}$, given the permutation $\mathbf{X}$. In this case, the topology of a graph is defined by a weighted adjacency matrix, $\mathbf{A} \in [0,1]^{n \times n}$, where each element, a_{ij}, indicates the soft connectivity between the i^{th} and the j^{th} nodes.

GM can be formulated as the classical QAP, that has been well studied in the literature of operation research since the $40's$. In operation research literature [15], Eq. 3 and Eq. 8 are known as Lawler's QAP [35] and Koopmans-Beckmann's QAP [26] respectively. Please refer to Fig. 3 for a taxonomy of previous GM works in computer vision from a QAP perspective. Roughly speaking, Lawler's QAP is more general than Koopmans-Beckmann's QAP because it has $\frac{1}{2}n_1^2 n_2^2$ free variables in $\mathbf{K}$ compared to $\frac{1}{2}n_1^2 + \frac{1}{2}n_2^2$ free variables in $\mathbf{A}_1$ and $\mathbf{A}_2$. In fact, Koopmans-Beckmann's QAP can always be represented as a special case of Lawler's QAP if we assume $\mathbf{K} = \mathbf{A}_2 \otimes \mathbf{A}_1$. In this particular Lawler's QAP, the edge feature has to be a 1-D scalar (*i.e.*, $\mathbf{q}_c = a_{ij}$) and the edge similarity has to be linear (*i.e.*, $\kappa_{c_1 c_2}^q = \langle \mathbf{q}_{c_1}^1, \mathbf{q}_{c_2}^2 \rangle$). This special QAP can be limited when representing the challenging matching problem encountered in real cases. However, Lawler's QAP considers more general similarity measures including Gaussian kernels, which take the linear similarity as an instance.

In general, it is unclear on how to understand the commonalities and differences between these two types of GMs. In the following, we will propose a simple and clean connection that exists using the proposed factorization. Observe that $\mathbf{K}_q$ can always be factorized

(*e.g.*, SVD) as $\mathbf{K}_q = \mathbf{U}\mathbf{V}^T$, where $\mathbf{U} \in \mathbb{R}^{m_1 \times c}$ and $\mathbf{V} \in \mathbb{R}^{m_2 \times c}$. Taking advantage of the low-rank structure of $\mathbf{K}_q$, Eq. 7 can be re-formulated[2] as follows:

$$\begin{aligned} J_{gm}(\mathbf{X}) &= \operatorname{tr}\left(\mathbf{K}_p^T\mathbf{X}\right) + \operatorname{tr}\left((\mathbf{U}\mathbf{V}^T)^T(\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2 \circ \mathbf{H}_1^T\mathbf{X}\mathbf{H}_2)\right) \\ &= \operatorname{tr}\left(\mathbf{K}_p^T\mathbf{X}\right) + \operatorname{tr}\left((\sum_{i=1}^{c}\mathbf{u}_i\mathbf{v}_i^T)^T(\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2 \circ \mathbf{H}_1^T\mathbf{X}\mathbf{H}_2)\right) \\ &= \operatorname{tr}\left(\mathbf{K}_p^T\mathbf{X}\right) + \sum_{i=1}^{c}\operatorname{tr}\left(\mathbf{A}_i^1\mathbf{X}\mathbf{A}_i^2\mathbf{X}^T\right), \end{aligned} \tag{9}$$

where $\mathbf{A}_i^1 = \mathbf{G}_1\operatorname{diag}(\mathbf{u}_i)\mathbf{H}_1^T \in \mathbb{R}^{n_1 \times n_1}$ and $\mathbf{A}_i^2 = \mathbf{G}_2\operatorname{diag}(\mathbf{v}_i)\mathbf{H}_2^T \in \mathbb{R}^{n_2 \times n_2}$ can be interpreted as adjacency matrices for the two graphs. Eq. 9 reveals that c, the rank of the edge affinity matrix $\mathbf{K}_q$, is a key characteristic of GM methods. If $c = 1$, Lawler's QAP downgrades to the Koopmans-Beckmann's QAP. In other cases when $c > 1$, solving Lawler's QAP can be cast as a joint optimization of c Koopmans-Beckmann's QAPs.

Despite its importance, the relation between Eq. 3 and Eq. 8 has been rarely explored in the literature of GM. In fact, many GM methods can benefit from this connection. Consider the special case when the node affinity matrix $\mathbf{K}_p$ is empty and the edge affinity matrix $\mathbf{K}_q = \mathbf{u}\mathbf{v}^T$ has a rank-1 structure. According to the factorization, we can conclude $\mathbf{K} = \mathbf{A}_2 \otimes \mathbf{A}_1$, where $\mathbf{A}_1 = \mathbf{G}_1\operatorname{diag}(\mathbf{u})\mathbf{H}_1^T$ and $\mathbf{A}_2 = \mathbf{G}_2\operatorname{diag}(\mathbf{v})\mathbf{H}_2^T$. In this case, the spectral matching algorithm [27] can be more efficiently computed as $\operatorname{eig}(\mathbf{K}) = \operatorname{eig}(\mathbf{A}_2) \otimes \operatorname{eig}(\mathbf{A}_1)$, where $\operatorname{eig}(\mathbf{K})$ denotes the leading eigen-vector of $\mathbf{K}$.

5.2 Connections between GM and ICP

To connect ICP with GM methods, we denote the node affinity matrix as, $\mathbf{K}_p(\mathcal{T}) \in \mathbb{R}^{n_1 \times n_2}$, where each element, $\kappa_{i_1 i_2}^p(\mathcal{T}) = -\|\mathbf{p}_{i_1}^1 - \tau(\mathbf{p}_{i_2}^2)\|_2^2$, encodes the negative Euclidean distance between nodes. Using this notation, the original objective (Eq. 5) can be concisely formulated as:

$$J_{icp}(\mathbf{X}, \mathcal{T}) = -\operatorname{tr}\left(\mathbf{K}_p(\mathcal{T})^T\mathbf{X}\right) + \psi(\mathcal{T}). \tag{10}$$

Eq. 10 reveals a clean connection between the ICP and GM objective (Eq. 7). Given the transformation ($\mathcal{T}$), minimizing J_{icp} over $\mathbf{X}$ is equivalent to maximize the node compatibility in Eq. 7. This optimization can be cast as a linear matching problem, which can be efficiently optimized by the Hungarian algorithm if $\mathbf{X}$ is a one-to-one mapping or the winner-take-all manner if $\mathbf{X}$ is a many-to-one mapping. In general, however, the joint optimization over $\mathbf{X}$ and $\mathcal{T}$ is non-convex, and no closed form solution is known. Typically, some sort of alternated minimization (*e.g.*, EM, coordinate-descent) is needed to find a local optima.

6 A PATH-FOLLOWING ALGORITHM

This section presents a path-following algorithm for approximating the GM problem. Traditionally, Eq. 3 is approached by a two-step scheme: (1) solving a continuously relaxed problem and (2) rounding the approximate solution to a binary one. This procedure has at least two limitations. First, the continuous relaxation is non-convex and prone to local optima. Second, the rounding step will inevitably cause accuracy loss because it is independent of the cost function. Inspired by [33], [59], we address these two issues using a path-following algorithm 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 to an integer one and therefore no rounding step is needed; (3) The iteratively updating procedure resembles the idea of numerical continuation methods [60], which have been used for solving nonlinear systems of equations for decades.

6.1 Convex and concave relaxation

To employ the path-following algorithm, we need to find a convex and concave relaxation of $J_{gm}(\mathbf{X})$. The factorization (Eq. 6) offers a simple and principled way to derive these approximations. To do that, let's first introduce an auxiliary function:

$$J_{con}(\mathbf{X}) = \sum_{i=1}^{c}\operatorname{tr}\left({\mathbf{A}_i^1}^T\mathbf{X}\mathbf{X}^T\mathbf{A}_i^1\right) + \operatorname{tr}\left(\mathbf{A}_i^2\mathbf{X}^T\mathbf{X}{\mathbf{A}_i^2}^T\right).$$

As we know, a permutation matrix[3] $\mathbf{X} \in \Pi$ is also an orthogonal one, satisfying $\mathbf{X}\mathbf{X}^T = \mathbf{X}^T\mathbf{X} = \mathbf{I}$. Therefore, $J_{con}(\mathbf{X}) = \gamma$ is always a constant,

$$\gamma = \sum_{i=1}^{c}\operatorname{tr}\left({\mathbf{A}_i^1}^T\mathbf{A}_i^1\right) + \operatorname{tr}\left(\mathbf{A}_i^2{\mathbf{A}_i^2}^T\right),$$

if $\mathbf{X}$ is a permutation matrix. Introducing into $J_{gm}(\mathbf{X})$ the constant term, $\frac{1}{2}J_{con}(\mathbf{X})$, yields two equivalent objectives:

$$\begin{aligned} J_{vex}(\mathbf{X}) &= J_{gm}(\mathbf{X}) - \frac{1}{2}J_{con}(\mathbf{X}) \\ &= \operatorname{tr}\left(\mathbf{K}_p^T\mathbf{X}\right) - \frac{1}{2}\sum_{i=1}^{c}\|\mathbf{X}^T\mathbf{A}_i^1 - \mathbf{A}_i^2\mathbf{X}^T\|_F^2, \end{aligned} \tag{11}$$

$$\begin{aligned} J_{cav}(\mathbf{X}) &= J_{gm}(\mathbf{X}) + \frac{1}{2}J_{con}(\mathbf{X}) \\ &= \operatorname{tr}\left(\mathbf{K}_p^T\mathbf{X}\right) + \frac{1}{2}\sum_{i=1}^{c}\|\mathbf{X}^T\mathbf{A}_i^1 + \mathbf{A}_i^2\mathbf{X}^T\|_F^2. \end{aligned} \tag{12}$$

The above two functions achieve the same value up to a constant difference $\frac{\gamma}{2}$ as $J_{gm}(\mathbf{X})$ with respect to any permutation and orthogonal matrix. Yet, they are supplementary to each other in the domain of doubly-stochastic matrices. In particular, the problems of maximizing $J_{vex}(\mathbf{X})$ and $J_{cav}(\mathbf{X})$ are convex and concave respectively. This is because their Hessians,

$$\nabla_{\mathbf{X}}^2 J_{vex}(\mathbf{X}) = -\sum_{i=1}^{c}(\mathbf{I}\otimes\mathbf{A}_i^1 - \mathbf{A}_i^2\otimes\mathbf{I})^T(\mathbf{I}\otimes\mathbf{A}_i^1 - \mathbf{A}_i^2\otimes\mathbf{I}),$$

$$\nabla_{\mathbf{X}}^2 J_{cav}(\mathbf{X}) = \sum_{i=1}^{c}(\mathbf{I}\otimes\mathbf{A}_i^1 + \mathbf{A}_i^2\otimes\mathbf{I})^T(\mathbf{I}\otimes\mathbf{A}_i^1 + \mathbf{A}_i^2\otimes\mathbf{I}),$$

are negative and positive semi-definite, respectively.

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

3. We need to introduce dummy selection variables if the two graphs are of different sizes.

6.2 A path-following strategy

The main challenge of approximating GM comes from its non-convex objective and the combinatorial constraints. Inspired by [33], [59], we optimize Eq. 3 by iteratively optimizing a series of the following convex-concave problems [61]:

$$\max_{\mathbf{X}\in\mathcal{D}} \quad J_\alpha(\mathbf{X}) = (1-\alpha)J_{vex}(\mathbf{X}) + \alpha J_{cav}(\mathbf{X}), \tag{13}$$

where $\alpha \in [0,1]$ is a tradeoff between the convex relaxation and the concave one.

The advantages of the path-following algorithm over conventional GM algorithms are three-fold: (1) The algorithm starts with a convex problem when $\alpha = 0$. Because the problem is convex, a numerical optimizer can find the global optimal solution independently of the initialization. (2) When $\alpha = 1$ (at the end of the iterations) the problem is concave. It has been well-studied [62] that any local optima of a concave optimization problem must lie at the extreme point of the constraint set. According to the Birkhoff-von Neumann theorem [63], all the doubly-stochastic matrices ($\mathbf{X} \in \mathcal{D}$) form the Birkhoff polytope, whose vertices are precisely the permutation matrices ($\mathbf{X} \in \Pi$). Therefore, the converged solution is guaranteed to be binary and no further discrete rounding step is needed. (3) By smoothly increasing α from 0 to 1, the path-following algorithm is more likely to find better local optima than gradient-based method.

To have a better understanding, we provide an example of optimizing GM using this strategy in Fig. 6. The graphs to be matched are extracted from one pair of car images used in the experiment (See Section 8.4). Fig. 6a plots the real GM objective (J_{gm}) and the objective (J_α) to be optimized with respect to the change of α. The convex (J_{vex}) and concave (J_{cav}) components are also plotted. Overall, four observations can be concluded looking at this figure. First, J_α equals to J_{vex} and J_{cav} when $\alpha = 0$ and $\alpha = 1$ respectively. Second, the curves of J_{vex} and J_{cav} are monotonically decreasing and increasing as $\alpha \to 1$. This is because as α increases, the concave part gets more control than the concvex part to direct the optimization of J_α. Third, the curve of J_α intersects J_{gm} at $\alpha = \frac{1}{2}$. This is true due to the fact that the following equation holds:

$$J_\alpha(\mathbf{X}) = J_{gm}(\mathbf{X}) + (\alpha - \frac{1}{2})J_{con}(\mathbf{X}), \tag{14}$$

by plugging Eq. 11 and Eq. 12 into Eq. 13. At last, when the algorithm converges at $\alpha = 1$, the gap between J_{gm} and the two relaxations is a constant $\frac{\gamma}{2}$. This is because $\mathbf{X}$ turns to be a permutation matrix (Fig. 6c) and the equation, $J_{con}(\mathbf{X}) = \gamma$, always holds at $\alpha = 1$.

6.3 Sub-problem optimization

For a specific α, we optimize $J_\alpha(\mathbf{X})$ using the Frank-Wolfe's algorithm (FW) [64], a simple yet powerful method for constrained nonlinear programming. Unlike gradient-based methods that require a projection onto the constraint, FW algorithm efficiently updates the solution by automatically staying in the feasible set.

Given an initial $\mathbf{X}_0$, FW successively updates the solution as $\mathbf{X}^* = \mathbf{X}_0 + \eta\mathbf{D}$ until converged. At each step, it needs to compute two components: the optimal direction $\mathbf{D} \in \mathcal{D}$ and the optimal step size $\eta \in [0,1]$. To compute $\mathbf{Y}$, we solve the following LP problem:

$$\max_{\mathbf{D}\in\mathcal{D}} \quad \mathrm{tr}\left(\nabla J_\alpha(\mathbf{X}_0)^T(\mathbf{D}-\mathbf{X}_0)\right), \tag{15}$$

whose objective is the first-order approximation of $J_\alpha(\mathbf{X})$ at the point $\mathbf{X}_0$. The gradients $\nabla J_\alpha(\mathbf{X}_0)$ can be efficiently computed using matrix operation as follows:

$$\nabla J_\alpha(\mathbf{X}) = \nabla J_{gm}(\mathbf{X}) + (\alpha - \frac{1}{2})\nabla J_{con}(\mathbf{X}),$$
$$\nabla J_{gm}(\mathbf{X}) = \mathbf{K}_p + \mathbf{H}_1(\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2 \circ \mathbf{K}_q)\mathbf{H}_2^T + \mathbf{G}_1(\mathbf{H}_1^T\mathbf{X}\mathbf{H}_2 \circ \mathbf{K}_q)\mathbf{G}_2^T,$$
$$\nabla J_{con}(\mathbf{X}) = 2\Big(\mathbf{G}_1(\mathbf{H}_1^T\mathbf{H}_1 \circ \mathbf{U}\mathbf{U}^T)\mathbf{G}_1^T + \mathbf{G}_2(\mathbf{H}_2^T\mathbf{H}_2 \circ \mathbf{V}\mathbf{V}^T)\mathbf{G}_2^T\Big)\mathbf{X}.$$

Similar to [33], [39], we adopt the Hungarian algorithm to efficiently solve this linear programming.

Once the gradient is computed, the line search for the optimal η can be found in closed form. The optimal η is the maximum point of the following parabola:

$$J_\alpha(\mathbf{X}_0 + \eta\mathbf{D}) = a\eta^2 + b\eta + \text{const}, \tag{16}$$

where a and b are the fixed 2^{nd} and 1^{st} order coefficients respectively. It is well-known that the optimum point of the parabola must be one of the three points, $\eta^* \in \{0, 1, -\frac{b}{2a}\}$. A straightforward way to obtain the optimal η^* is to compute $J_\alpha(\mathbf{X}_0 + \eta\mathbf{D})$ at each point and pick the one yielding the largest value. In fact, it is more efficient to choose η^* based on the geometry of the parabola. Fig. 7 illustrates the eight possible configurations of the parabola and the corresponding optimal η.

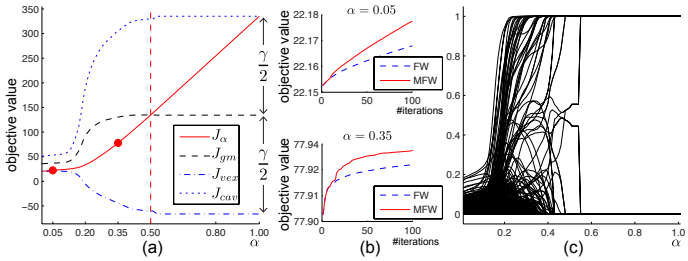

Fig. 6. An example of using the path-following algorithm for optimizing the GM problem. (a) The comparison of the objectives during the optimization with respect to the change of α. (b) The comparison of using FW and MFW for optimizing $J_\alpha(\mathbf{X})$ at two αs. (c) The change of x_{ij} (each curve) as $\alpha \to 1$.

6.4 Other implementation details

A similar path-following strategy was proposed in [33] and its performance over the state-of-the-art methods has been therein demonstrated for solving the less general GM problem (Eq. 8). After exhaustive testing this strategy for solving the most general GM problem (Eq. 3), we empirically found that results can be improved using the following two 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

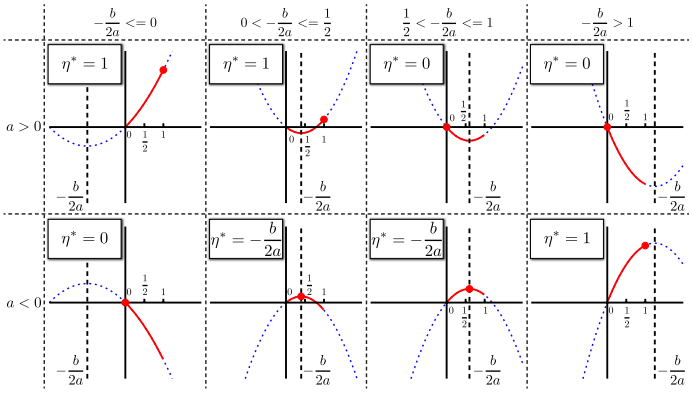

Fig. 7. Eight possible shapes of the parabola $f(\eta) = a\eta^2 + b\eta$ and the corresponding optimal step sizes $\eta^* = \arg\max_{\eta \in [0,1]} f(\eta)$.

Algorithm 1: A path-following algorithm

input : $\mathbf{K}_p$, $\mathbf{K}_q$, $\mathbf{G}_1$, $\mathbf{H}_1$, $\mathbf{G}_2$, $\mathbf{H}_2$, δ, $\mathbf{X}_0$
output: $\mathbf{X}$

Initialize $\mathbf{X}$ *to be a doubly stochastic matrix;*
Factorize $\mathbf{K}_q = \mathbf{U}\mathbf{V}^T$;
for $\alpha = 0 : \delta : 1$ **do** *Path-following*
  if $\alpha = 0.5$ *and* $J_{gm}(\mathbf{X}) < J_{gm}(\mathbf{X}_0)$ **then**
    Update $\mathbf{X} \leftarrow \mathbf{X}_0$;
  Optimize Eq. 13 via MFW to obtain $\mathbf{X}^*$;
  if $J_{gm}(\mathbf{X}^*) < J_{gm}(\mathbf{X})$ **then**
    Optimize Eq. 3 via one step of FW to obtain $\mathbf{X}^*$;
  Update $\mathbf{X} \leftarrow \mathbf{X}^*$;

Frank-Wolfe (MFW) [65] to find a better searching direction $\mathbf{D}$ by a convex combination of previously solutions. Fig. 6b compares MFW with the FW for two αs. As we can see, MFW converges much faster than FW.

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 solution 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 $J_{gm}(\mathbf{X})$. It is possible that $J_\alpha(\mathbf{X})$ increases while $J_{gm}(\mathbf{X})$ decreases. To escape from this phenomenon, we keep increasing the global score of $J_{gm}(\mathbf{X})$ 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 $J_{gm}(\mathbf{X})$.

Algorithm 1 summarizes the work-flow of our algorithm. The initial $\mathbf{X}$ can be an arbitrary doubly stochastic matrix. The complexity of our algorithm can be roughly calculated as $O\big(T(\tau_{hun} + \tau_\nabla + \tau_\lambda) + \tau_{svd}\big)$, where T is the number of iterations for the FW, and $\tau_{svd} = m_1 m_2^2$ is the cost of computing the SVD of $\mathbf{K}_q$. 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, $\tau_\nabla = \tau_\lambda = m_1 m_2$.

7 Deformable Graph Matching

GM has been widely used as a general tool in matching point sets with similar structures. In many computer vision applications, it is often required that the matching between two sets is constrained by a geometric transformation. It is unclear how to incorporate geometric constraints into GM methods. This section describes deformable graph matching (DGM), that incorporates rigid and non-rigid transformations into graph matching.

7.1 Objective function

To simplify the discussion and to be consistent with ICP, we compute the node feature of each graph $\mathcal{G} = \{\mathbf{P}, \mathbf{Q}, \mathbf{G}, \mathbf{H}\}$ simply as the node coordinates, $\mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_n] \in \mathbb{R}^{d \times n}$. Similarly, the edge features $\mathbf{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_m] \in \mathbb{R}^{d \times m}$ are computed as the coordinate difference between the connected nodes, *i.e.*, $\mathbf{q}_c = \mathbf{p}_i - \mathbf{p}_j$, where $g_{ic} = h_{jc} = 1$. In this case, the edge feature can be conveniently computed in a matrix form as, $\mathbf{Q} = \mathbf{P}(\mathbf{G} - \mathbf{H})$.

Suppose that we are given two graphs, $\mathcal{G}_1 = \{\mathbf{P}_1, \mathbf{Q}_1, \mathbf{G}_1, \mathbf{H}_1\}$ and $\mathcal{G}_2 = \{\mathbf{P}_2, \mathbf{Q}_2, \mathbf{G}_2, \mathbf{H}_2\}$, and a geometrical transformation defined on points by $\tau(\cdot)$. Similar to ICP, we compute the node affinity $\mathbf{K}_p(\mathcal{T}) \in \mathbb{R}^{n_1 \times n_2}$ and the edge affinity $\mathbf{K}_q(\mathcal{T}) \in \mathbb{R}^{m_1 \times m_2}$ as linear functions of the Euclidean distance, *i.e.*:

$$\begin{aligned} \kappa^p_{i_1 i_2}(\mathcal{T}) &= -\|\mathbf{p}^1_{i_1} - \tau(\mathbf{p}^2_{i_2})\|_2^2, \\ \kappa^q_{c_1 c_2}(\mathcal{T}) &= \beta - \|\underbrace{(\mathbf{p}^1_{i_1} - \mathbf{p}^1_{j_1})}_{\mathbf{q}^1_{c_1}} - \underbrace{(\tau(\mathbf{p}^2_{i_2}) - \tau(\mathbf{p}^2_{j_2}))}_{\tau(\mathbf{q}^2_{c_2})}\|_2^2, \end{aligned} \quad (17)$$

where β is chosen to be reasonably large to ensure that the pairwise affinity is greater than zero.

Recall that the factorization (Eq. 6) reveals that the goal of GM (Eq. 7) is similar to ICP (Eq. 10). In order to make GM more robust to geometric deformations, DGM aims to find the optimal correspondence $\mathbf{X}$ as well as the optimal transformation $\mathcal{T}$ such that the global consistency can be maximized:

$$\max_{\mathbf{X} \in \Pi, \mathcal{T} \in \Psi} J_{dgm}(\mathbf{X}, \mathcal{T}) = \operatorname{tr}\left(\mathbf{K}_p(\mathcal{T})^T \mathbf{X}\right) + \lambda \operatorname{tr}\left(\mathbf{K}_q(\mathcal{T})^T \mathbf{Y}\right) - \psi(\mathcal{T}), \quad (18)$$

where $\lambda \geq 0$ is used to balance between the importance of the node and edge consistency. Similar to ICP, $\psi(\mathcal{T})$ and Ψ are used to constrain and penalize the transformation parameter. Eq. 18 extends ICP by adding the transformation. In particular, if $\lambda = 0$, solving DGM is equivalent to ICP. In other case when $\lambda > 0$ and $\mathcal{T}$ is known, solving DGM is identical to a GM problem.

7.2 Optimization

Due to the non-convex nature of the objective, we optimize J_{dgm} by alternatively solving the correspondence ($\mathbf{X}$) and the transformation parameter ($\mathcal{T}$). The initialization is important for the performance of DGM. However,

how to select a good initialization is beyond the scope of this paper and we simply set the initial transformation as an identity one, *i.e.*, $\tau(\mathbf{p}) = \mathbf{p}$.

Optimizing Eq. 18 consists of alternation between optimizing for the correspondence and the geometric transformation. Given the transformation $\mathcal{T}$, DGM is equivalent to a traditional GM problem. To find the node correspondence $\mathbf{X}$, we adopt the path-following algorithm by optimizing Eq. 13. Given the correspondence matrix $\mathbf{X}$, the optimization over the transformation parameter $\mathcal{T}$ is similar to ICP. After some linear algebra, it can be shown that for the following transformations, the parameter can be computed in closed-form.

Similarity transformation: According to the definition in Eq. 17, the similarity transformation of the edge feature is $\tau(\mathbf{q}) = s\mathbf{R}\mathbf{q}$ or $\tau(\mathbf{Q}) = s\mathbf{R}\mathbf{Q}$. Since the translation $\mathbf{t}$ only affects the node feature, the optimal $\mathbf{t}^*$ can be computed as follows once the optimal scalar s^* and rotation $\mathbf{R}^*$ are known:

$$\mathbf{t}^* = \bar{\mathbf{p}}_1 - s^*\mathbf{R}^*\bar{\mathbf{p}}_2, \text{ where } \bar{\mathbf{p}}_1 = \frac{\mathbf{P}_1\mathbf{X}\mathbf{1}_{n_2}}{\mathbf{1}_{n_1}^T\mathbf{X}\mathbf{1}_{n_2}}, \ \bar{\mathbf{p}}_2 = \frac{\mathbf{P}_2\mathbf{X}^T\mathbf{1}_{n_1}}{\mathbf{1}_{n_1}^T\mathbf{X}\mathbf{1}_{n_2}}.$$

After centerizing the data, $\bar{\mathbf{P}}_1 = \mathbf{P}_1 - \bar{\mathbf{p}}_1\mathbf{1}_{n_1}^T$ and $\bar{\mathbf{P}}_2 = \mathbf{P}_2 - \bar{\mathbf{p}}_2\mathbf{1}_{n_2}^T$, the optimal rotation and scaling can be achieved at:

$$\mathbf{R}^* = \mathbf{U}\,\mathrm{diag}(1, \cdots, |\mathbf{U}\mathbf{V}^T|)\mathbf{V}^T,$$
$$s^* = \frac{\mathrm{tr}(\mathbf{\Sigma})}{\mathrm{tr}\left(\mathbf{1}_{n_1\times 2}(\bar{\mathbf{P}}_2 \circ \bar{\mathbf{P}}_2)\mathbf{X}^T\right) + \lambda_q\,\mathrm{tr}\left(\mathbf{1}_{m_1\times 2}(\mathbf{Q}_2 \circ \mathbf{Q}_2)\mathbf{Y}^T\right)},$$

where $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \bar{\mathbf{P}}_1\mathbf{X}\bar{\mathbf{P}}_2^T + \lambda_q\mathbf{Q}_1\mathbf{Y}\mathbf{Q}_2^T$.

Affine transformation: After applying the transformation, the edge feature becomes $\tau(\mathbf{q}) = \mathbf{V}\mathbf{q}$ or $\tau(\mathbf{Q}) = \mathbf{V}\mathbf{Q}$. Similar to the similarity transformation, the optimal translation is dependent on the optimal affine matrix $\mathbf{V}^*$:

$$\mathbf{t}^* = \bar{\mathbf{p}}_1 - \mathbf{V}^*\bar{\mathbf{p}}_2, \text{ where } \bar{\mathbf{p}}_1 = \frac{\mathbf{P}_1\mathbf{X}\mathbf{1}_{n_2}}{\mathbf{1}_{n_1}^T\mathbf{X}\mathbf{1}_{n_2}}, \ \bar{\mathbf{p}}_2 = \frac{\mathbf{P}_2\mathbf{X}^T\mathbf{1}_{n_1}}{\mathbf{1}_{n_1}^T\mathbf{X}\mathbf{1}_{n_2}}.$$

After centerizing all the data, we can compute the optimal affine transformation:

$$\mathbf{V}^* = \mathbf{A}\mathbf{B}^{-1}, \text{ where } \mathbf{A} = \bar{\mathbf{P}}_1\mathbf{X}\bar{\mathbf{P}}_2^T + \lambda_q\mathbf{Q}_1\mathbf{Y}\mathbf{Q}_2^T,$$
$$\mathbf{B} = \bar{\mathbf{P}}_2\,\mathrm{diag}(\mathbf{X}^T\mathbf{1}_{n_1})\bar{\mathbf{P}}_2^T + \lambda_q\mathbf{Q}_2\,\mathrm{diag}(\mathbf{Y}^T\mathbf{1}_{m_1})\mathbf{Q}_2^T.$$

Non-rigid transformation: According to the definition in Eq. 17, the non-rigid transformation of the edge features is $\tau(\mathbf{q}_c) = \mathbf{q}_c + \mathbf{W}(\phi(\mathbf{p}_i) - \phi(\mathbf{p}_j))$, where edge $\mathbf{q}_c$ connects between point $\mathbf{p}_i$ and $\mathbf{p}_j$. In matrix form, we can show that $\tau(\mathbf{Q}) = \mathbf{Q} + \mathbf{W}\mathbf{L}_q$, where $\mathbf{L}_q = \mathbf{L}_p(\mathbf{G} - \mathbf{H})$. Setting the gradient of $J_{dgm}(\mathbf{W})$ to zero yields the optimal weight matrix $\mathbf{W}$ as $\mathbf{W}^* = \mathbf{A}\mathbf{B}^{-1}$, where

$$\mathbf{A} = \mathbf{P}_1\mathbf{X} - \mathbf{P}_2\,\mathrm{diag}(\mathbf{X}^T\mathbf{1}_{n_1}) + \lambda_q\mathbf{Q}_1\mathbf{Y}(\mathbf{G}_2 - \mathbf{H}_2)^T$$
$$- \lambda_q\mathbf{Q}_2\,\mathrm{diag}(\mathbf{Y}^T\mathbf{1}_{m_1})(\mathbf{G}_2 - \mathbf{H}_2)^T,$$
$$\mathbf{B} = \mathbf{L}_p\,\mathrm{diag}(\mathbf{X}^T\mathbf{1}_{n_1}) + \lambda_w\mathbf{I}_{n_2} + \lambda_q\mathbf{L}_q\,\mathrm{diag}(\mathbf{Y}^T\mathbf{1}_{m_1})(\mathbf{G}_2 - \mathbf{H}_2)^T.$$

7.3 Comparison with ICP

It is well known that the performance of ICP algorithms largely depends on the effectiveness of the initialization step. In the following example, we empirically illustrate how by adding additional pairwise constrains, DGM is less sensitive to the initialization. Fig. 8a illustrates the problem of aligning two fish shapes under varying values for the initial rotation and scale parameters. As shown in Fig. 8b, ICP gets trapped into a local optima if the orientation gap is larger than $\frac{1}{3}\pi$ (the error should be 0). Similarly, DGM fails for large orientation gap after two iterations (the left column of Fig. 8c). However, as the number of iterations increasing, DGM is able to match shapes with very large deformation in rotation and scales. After 24 iterations, DGM ultimately finds the optimal matching for all the initializations (the right column of Fig. 8c). This experiment shows that adding pairwise constraints can make the ICP algorithm more robust to the problem of local optima.

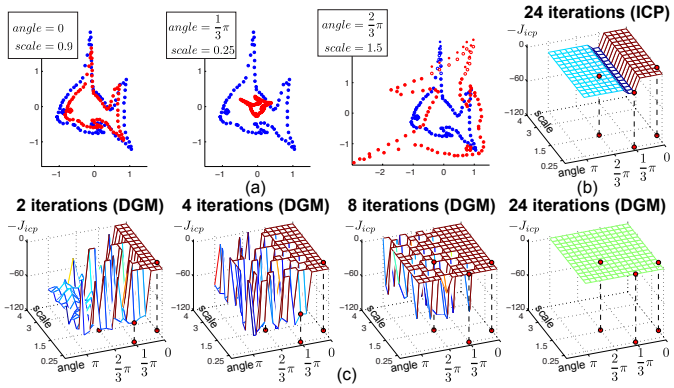

Fig. 8. Comparison between ICP and DGM for aligning shapes for several initial values of rotation and scale parameters. (a) Examples of initializations for different angles and scales. (b) Objective surfaces obtained by ICP. (c) Objective surfaces obtained by DGM.

8 Experiments

This section reports experimental results on four benchmark datasets and compares FGM to several state-of-the-art methods for GM and ICP. In the first three experiments, we test the performance of using the path-following algorithm for optimizing GM problems. In the last experiment we add a known geometrical transformation between graphs and compare with ICP on the problem of matching rigid and non-rigid shapes.

8.1 Baselines and evaluation metrics

In the experiment, we compared against 8 algorithms.

Hungarian algorithm (HUN): HUN [16] is the algorithm that solves the linear assignment problem in polynomial time. We use it as a baseline to find the correspondence $\mathbf{X}$ that maximize the 1^{st}-order term of GM's objective: $\mathrm{tr}(\mathbf{K}_p\mathbf{X}^T)$.

Graduated assignment (GA): GA [36] performs gradient ascent on a relaxation of Eq. 3 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 [27] optimizes a relaxed problem of Eq. 3 that drops the affine constraints and introduces a unit-length constraint on $\mathbf{X}$, that is:

$$\max_{\mathbf{X}} J_{gm}(\mathbf{X}), \quad \text{s.t. } \|\operatorname{vec}(\mathbf{X})\|_2^2 = 1.$$

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

Spectral matching with affine constraints (SMAC): SMAC [28] adds affine constraints to SM maximizing:

$$\max_{\mathbf{X}} J_{gm}(\mathbf{X}), \quad \text{s.t. } \mathbf{A}\operatorname{vec}(\mathbf{X}) = \mathbf{b} \text{ and } \|\operatorname{vec}(\mathbf{X})\|_2^2 = 1.$$

The solution is also an eigenvalue problem.

Integer projected fixed point method (IPFP): IPFP [39] can take any continuous or discrete solution as inputs and iteratively improve the solution. In our experiments, we implemented two versions: (1) IPFP-U, which starts from the same initial $\mathbf{X}$ as our method; (2) IPFP-S, which is initialized by SM.

Probabilistic matching (PM): PM [42] designs $\min_{\mathbf{X}\in\mathcal{D}} S(\mathbf{Y}\|\mathbf{X})$, a convex objective function that can be globally optimized, where $S(\cdot)$ denotes the relative entropy based on the Kullback-Leibler divergence and $\mathbf{Y}$ is calculated by marginalizing $\mathbf{K}$.

Re-weighted random walk matching (RRWM): RRWM [43] 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.

Concave optimization (CAV): Concave optimization was firstly used in [59] for optimizing GM. Compared to conventional method, concave optimization is guaranteed to converge at an integer solution and no further rounding step is needed. Similar to [59], we implemented CAV by employing the Frank-Wolfe algorithm to iteratively optimizing the concave relaxation (Eq. 12) of GM. To demonstrate the benefit of the path-following procedure, CAV was initialized by solving the same convex relaxation (Eq. 11) as FGM-D. The main difference between CAV and FGM-D is that the latter employs a path-following strategy to iteratively improve the result.

Degenerate similiary (DEN): To evaluate the effect of each component in the factorization (Eq. 6), we generated a degenerate edge similarity by setting $\mathbf{K}_q = \mathbf{1}$ so that the edge similarly is only dependent on the topology of the graph defined by $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{H}_1$ and $\mathbf{H}_2$. DEN was optimized using the same path-following method.

Factorized graph matching (FGM): We implemented two versions of our method, FGM-U for undirected graphs with only symmetric edge features and FGM-D for directed graphs with either symmetric or asymmetric edge features. FGM-U was first proposed in [17], which contains a similar factorization to Eq. 6. The main difference between FGM-U and FGM-D is that the former factorization can only be used for undirected graphs, while FGM-D can handle both undirected and directed graphs. To use FGM-U for matching graphs with directed edges, we converted the directed edges to undirected edges and computed the edge affinity as the average value of the original affinity between the directed edges. The parameters were fixed to $\delta = 0.01$.

We used codes from the author's websites for all methods. The code was implemented in Matlab on a laptop platform with 2.4G Intel CPU and 4G memory. FGM-U and FGM-D were 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, $acc = \operatorname{tr}(\mathbf{X}_{alg}^T\mathbf{X}_{tru})/\operatorname{tr}(\mathbf{1}_{n_2\times n_1}\mathbf{X}_{tru})$, 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, $obj = J_{gm}(\mathbf{X}_{alg})/J_{gm}(\mathbf{X}_{ours})$ is computed as the ratio between the objective values of FGM-D and other algorithms.

8.2 Synthetic dataset

This experiment performed a comparative evaluation of GM algorithms on randomly synthesized graphs following the experimental protocol of [28], [36], [43]. 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 was 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 was perturbed by adding a random Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Notice that the edge feature was asymmetrical. The edge affinity $\mathbf{K}_q$ was computed as $k^q_{c_1c_2} = \exp(-(q^1_{c_1} - q^2_{c_2})^2/0.15)$ 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 ratio. In the first setting (Fig. 9a), 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. 9b), we perturbed the edge weights by changing the noise parameter σ from 0 to 0.2, while fixing the other two parameters $n_{out} = 0$ and $\rho = 1$. In the last case (Fig. 9c), we verified the performance of matching sparse graphs by varying ρ from 1 to 0.3. In most cases, FGM-D achieved the best performance over all other algorithms in terms of both accuracy and objective ratio. RRWM and CAV are our closest competitors. FGM-U did not achieve comparatively good results because it solved an undirected approximation of the original problem. Initialized by the same convex solution as FGM-D, however, CAV performed worse than FGM-D due to the lack of the robust path-following strategy. We also found by only relying on the graph topology ($\mathbf{K}_q = \mathbf{1}$), DEN would perform dramatically worse than other baselines even using the path-following algorithm. This indicates the importance of designing a good edge similarity for the problem.

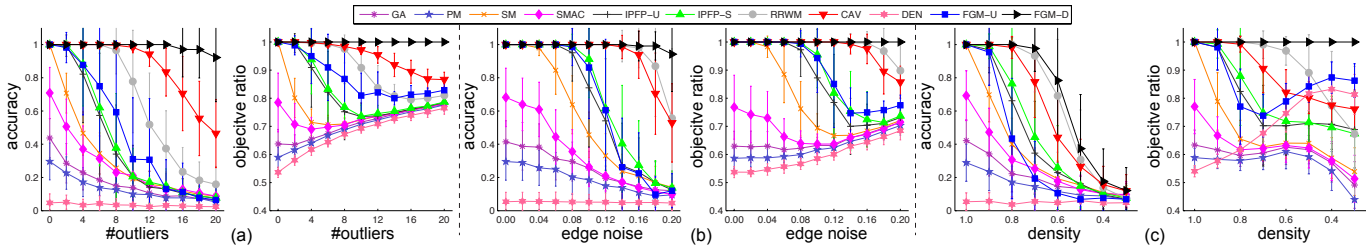

Fig. 9. Comparison of GM methods on synthetic datasets. (a) Performance as a function of the outlier number (n_{out}). (b) Performance as a function of the edge noise (σ). (c) Performance as a function of the density of edges (ρ).

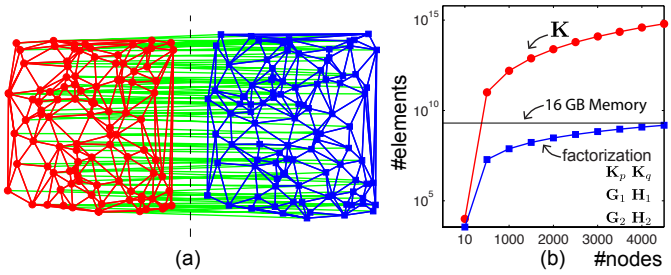

Fig. 10. Memory cost for GM problems. (a) An example of pair of graphs with 100 nodes. The edge connection is computed using Delaunay triangulation. (b) Number of elements to represent the affinity matrix.

FGM provides a superior matching performance. In addition, FGM uses less memory to represent the affinity matrix. Fig. 10a plots two random graphs with 100 nodes. We computed the graph connection using Delaunay triangulation. Fig. 10b compares the number of elements in the original affinity matrix $\mathbf{K}$ with the six smaller components. The proposed factorization allows to represent large-scale GM with much less memory.

8.3 House image dataset

The CMU house image sequence[4] was commonly used to test the performance of graph matching algorithms [40], [43], [45], [46]. This dataset consists of 111 frames of a house, each of which has been manually labeled with 30 landmarks. We used Delaunay triangulation to connect the landmarks. The edge weight q_c was computed as the pairwise distance between the connected nodes. Due to the symmetry of distance-based feature, the edge was undirected. Given an image pair, the edge-affinity matrix $\mathbf{K}_q$ was computed by $k^q_{c_1c_2} = \exp(-(q^1_{c_1} - q^2_{c_2})^2/2500)$ 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 by $0 : 10 : 90$ frames and computed the average matching accuracy and objective ratio per sequence gap. Fig. 11a demonstrates an example pair of two frames.

We tested the performance of GM methods under two scenarios. In the first case (Fig. 11b) we used all 30 nodes (*i.e.*, landmarks) and in the second one (Fig. 11c) we matched sub-graphs by randomly picking 25 landmarks. First of all, DEN achieved the worst performance, because it only relies on the graph topology to find the correspondence. Secondly, IPFP-S, RRWM, CAV, FGM-U and FGM-D almost obtained perfect matching of the original graphs in the first case (Fig. 11b). As some nodes became invisible and the graph got corrupted (Fig. 11c), the performance of all the methods degraded. However, FGM-U and FGM-D consistently achieved the best performance. The results demonstrate the advantages of the path-following algorithm over other state-of-the-art methods in solving general GM problems. In addition, it is interesting to notice that FGM-U and FGM-D performed similarly in both cases. This is because FGM-U can be considered as a special case of FGM-D when the graph consists of undirected edges.

8.4 Car and motorbike image dataset

The car and motorbike image dataset was created in [44]. This dataset consists of 30 pairs of car images and 20 pairs of motorbike images taken from the PASCAL challenges. Each pair contains $30 \sim 60$ ground-truth correspondences. We computed for each node the feature, p_i, as the orientation of the normal vector to the contour. We adopted the Delaunay triangulation to build the graph. In this experiment, we considered the most general graph where the edge was directed and the edge feature was asymmetrical. More specifically, each edge was represented by a couple of values, $\mathbf{q}_c = [d_c, \theta_c]^T$, where d_c was the pairwise distance between the connected nodes and θ_c was the 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 $k^q_{c_1c_2} = \exp(-\sigma|d_{c_1} - d_{c_2}| - (1-\sigma)|\theta_{c_1} - \theta_{c_2}|)$, where $\sigma = [0, 1]$ is the parameter that is a trade-off between the distance and angle features. We set $\sigma = \frac{1}{2}$ in the experiments to provide the same importance to the two edge features. Fig. 12a and Fig. 12b 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. Similarly, we compared FGM-D against 11 state-of-the-art methods. However, we were unable to directly use FGM-U to match directed graphs. Therefore, we ran FGM-U on

4. http://vasc.ri.cmu.edu//idb/html/motion/house

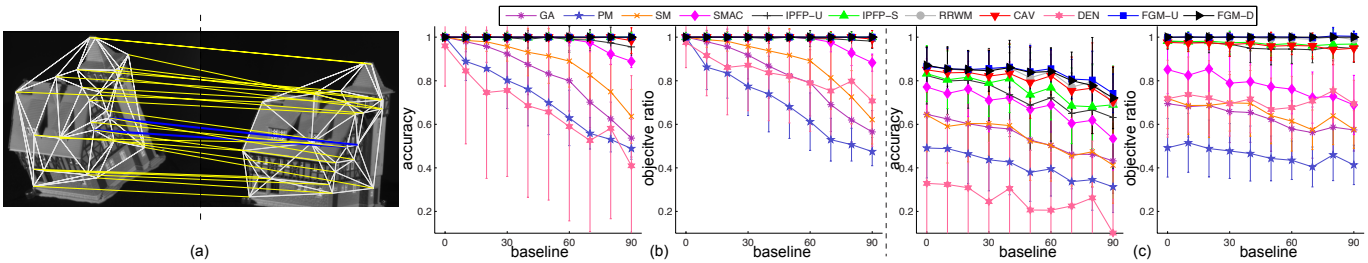

Fig. 11. Comparison of GM methods on the CMU house datasets. (a) An example pair of frames with the correspondence generated by our method, where the blue lines indicate incorrect matches. (b) Performance of several algorithms using 30 nodes. (c) Performance using 25 nodes.

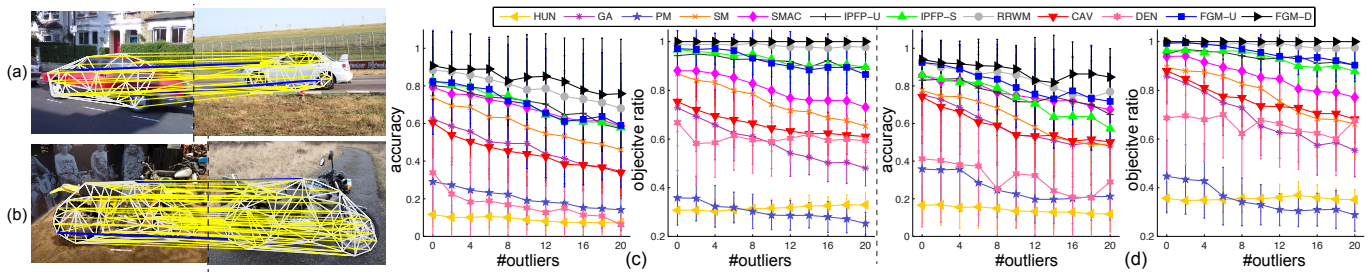

Fig. 12. Comparison of GM methods for the car and motorbike dataset. (a) An example pair of car images with the correspondence generated by our method, where the blue lines indicate incorrect matches. (b) An example pair of motorbike images. (c) Performance as a function of the outlier number for the car images. (d) Performance as a function of the outlier number for the motorbike images.

an approximated undirected graph, where we computed the feature of the undirected edge as the average value of the original affinities between the directed edges.

As observed in Fig. 12c-d, only using the node similarity (HUN) or a degenerate edge similarity (DEN) cannot perform well on this challenging dataset. This indicates the importance of designing good similarity functions to solve matching problem in realistic datasets. In addition, the proposed FGM-D consistently outperformed other methods on both datasets. Based on similar path-following strategy, however, FGM-U was unable to achieve the same accuracy because the graph consisted of directed edges and FGM-U was only applicable to undirected edges. It is worth to point that CAV performed not very well compared to FGM-D although it solved the same initial convex problem and optimized the same concave relaxation at the final step. This result as well as the previous experiment clearly demonstrated the path-following algorithm used by FGM-D provided a better optimization strategy than existing approaches. Finally, it is important to remind the reader that without the factorization proposed in this work it is not possible to apply the path-following method to general graphs.

Fig. 13a-b evaluate the performance of GM methods on the car and motorbike datasets for different weights (σ) between the distance and angle edge features. When $\sigma = 0$, the edge similarity is solely dependent on the angle feature. On the other hand, when $\sigma = 1$, the edge similarity is computed from edge distances. It can be observed that for most methods, the best performance is

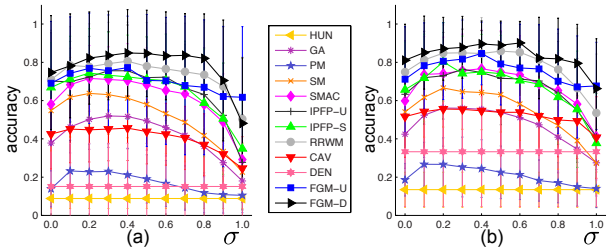

Fig. 13. Comparison of GM methods using different feature weights (σ) on the car (a) and motorbike (b) datasets.

achieved around $\sigma = 0.5$. This indicates a combination of distance and angle features provides a better similarity for this challenging task. In addition, our method (FGM-D) consistently achieved the best performance compared to others for most settings. Interestingly, when σ is close to 1 and the similarity is becoming a symmetric one (using distance features), our un-directed method (FGM-U) slightly out-performed the more general FGM-D.

8.5 Fish and character shape dataset

The UCF shape dataset [53] has been widely used for comparing ICP algorithms. In our experiment, we used two different templates. The first one has 91 points sampled from the outer contour of a tropical fish. The second one consist of 105 points sampled from a Chinese character. For each template, we designed two series of experiments to measure the robustness of an algorithm

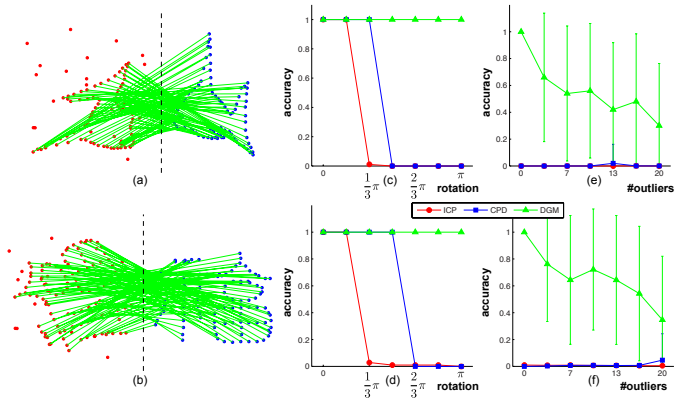

Fig. 14. Comparison between DGM and ICP on the UCF shape datasets. (a-b) Two example pairs of shapes aligned using DGM. The red shape (left) is a rotated version of the blue one (right) by $\frac{2}{3}\pi$ and 20 random outliers were added. (c-d) Matching performance as a function of the initial rotations. (e-f) Matching performance as a function of the number of outliers.

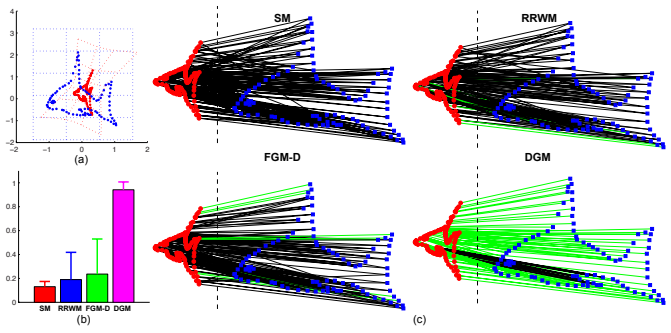

Fig. 15. Comparison between DGM and GM methods for aligning non-rigidly deformed shapes. (a) An example of two fishes, where the red one is generated by a non-rigid transformation from the blue one. (b) Accuracy. (c) Results, where the green and black lines indicate correct and incorrect correspondence respectively.

under different deformations and outliers. In the first series of experiments, we rotated the template with a varying degree (between 0 and π). In the second set of experiments, a varying amount of outliers (between 0 and 20) were randomly added into the bounding box of template. For instance, Fig. 14a-b illustrate two pairs of example shapes with 20 outliers. We repeated the random generation 50 times for different levels of noise and compared DGM with ICP and the coherent point drifting (CPD) [48]. The ICP algorithm was implemented by ourselves and CPD implementation was taken from the authors' website. We initialized all the algorithms with the same transformation, *i.e.*, $\tau(\mathbf{p}) = \mathbf{p}$. In DGM, Delaunay triangulation was employed to compute the graph structure. Recall that DGM simultaneously computes the correspondence and the rotation.

As shown in Fig. 14c-d, the proposed DGM can perfectly match the shapes across all the rotations without outliers, whereas both ICP and CPD get trapped in the local optimal when the rotation is larger than $\frac{2}{3}\pi$. When the number of outliers increases, DGM can still match most points under large rotation at $\frac{2}{3}\pi$. In contrast, ICP and CPD drastically failed in presence of outliers and large rotations (Fig. 14e-f). In addition to a similarity transform, DGM can also incorporate non-rigid transformations in GM. Similar to the rigid case described in the main submission, we synthesized the non-rigid shape from the UCF shape dataset [53]. To generate the nonrigid transformation, we followed a similar setting in [48], where the domain of the point set was parameterized by a mesh of control points. The deformation of the mesh was modeled as an spline-based interpolation of the perturbation of the control points. We repeated the random generation 50 times. For instance, Fig. 15a illustrates a synthetic pair of graphs.

We compared DGM with other two state-of-the-art GM methods: SM [27] and RRWM [43]. In addition, we tested the performance of our algorithm (FGM-D) only using the path-following algorithm for computing the correspondence but without estimating the transformation. As shown in Fig. 15b-c, FGM-D performed better than the other two GM methods. This is due to the path-following algorithm that is more accurate in optimizing GM problems. In addition, DGM significantly improved FGM-D by estimating the transformation.

9 Conclusions and Future Work

This paper proposes FGM, a new framework for understanding and optimizing GM problems. The key idea is a novel factorization of the pairwise affinity matrix into six smaller components. Four main benefits follow from factorizing the affinity matrix. First, there is no need to explicitly compute the costly affinity matrix. Second, it allows for a unification of GM methods as well as existing ICP algorithms. Third, the factorization enables the use of path-following algorithms that improve the performance of GM methods. Finally, it becomes easier to incorporate global geometrical transformation in GM.

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 exploring the extension of FGM to other higher-order graph matching problems [42], [46] as well as learning parameters for graph matching [44], [45].

Beyond GM problems, there are other computer vision problems using pairwise constraints that can benefit for the proposed factorization. A popular problem is segmentation using Markov random fields (MRFs) [66]. Given n_1 nodes and n_2 labels, MRF problems consist in finding the optimal labeling matrix $\mathbf{X} \in \Phi$ that defines a many-to-one mapping from nodes to labels, *i.e.*:

$$\Phi = \{\mathbf{X} | \mathbf{X} \in \{0,1\}^{n_1 \times n_2}, \mathbf{X}\mathbf{1}_{n_2} = \mathbf{1}_{n_1}\}. \quad (19)$$

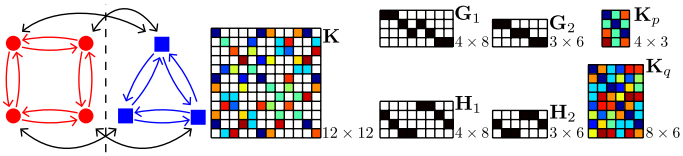

Fig. 16. An MRF example of four nodes (red circles) with three labels (blue boxes). Similar to GM, the affinity matrix $\mathbf{K}$ can be factorized into six smaller matrices.

Fig. 16 illustrates a simple MRF problem, where four nodes need to be assigned to three labels. Under the many-to-one constraint (Eq. 19), MRF can be formalized as a similar QAP as follows:

$$\max_{\mathbf{X}\in\Phi} \quad J_{mrf}(\mathbf{X}) = \mathrm{vec}(\mathbf{X})^T \mathbf{K}\, \mathrm{vec}(\mathbf{X}).$$

Although it is originally designed for GM problems, our factorization (Eq. 6) can be applied to model and understand MRF problems as well. For instance, the 12-by-12 $\mathbf{K}$ modeling the problem shown in Fig. 16 can be decomposed into six components. In particular, $\mathbf{G}_1$ and $\mathbf{H}_1$ are two 4-by-8 binary matrices, encoding the node-edge incidence of the node graph. $\mathbf{G}_2$ and $\mathbf{H}_2$ are two 3-by-6 binary ones, describing the fully connected label graph. $\mathbf{K}_p$ and $\mathbf{K}_q$ are used to encode the first-order and second-order potentials respectively. In future work, we plan to further explore this connection.

Acknowledgments This work was partially supported by the National Science Foundation under Grant No. EEEC-0540865, RI-1116583 and CPS-0931999. Any opinions, findings, and conclusions expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] H. Jiang, S. X. Yu, and D. R. Martin, "Linear scale and rotation invariant matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 7, pp. 1339–1355, 2011.
[2] R. Szeliski, *Computer vision: algorithms and applications*. Springer, 2010.
[3] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *ICCV*, 2005.
[4] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[5] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.
[6] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *CVPR*, 2005.
[7] D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin, and E. Boyer, "Articulated shape matching using Laplacian eigenfunctions and unsupervised point registration," in *CVPR*, 2008.
[8] M. Leordeanu, M. Hebert, and R. Sukthankar, "Beyond local appearance: Category recognition from pairwise interactions of simple features," in *CVPR*, 2007.
[9] O. Duchenne, A. Joulin, and J. Ponce, "A graph-matching kernel for object categorization," in *ICCV*, 2011.
[10] J. Hays, M. Leordeanu, A. A. Efros, and Y. Liu, "Discovering texture regularity as a higher-order correspondence problem," in *ECCV*, 2006.
[11] W. Brendel and S. Todorovic, "Learning spatiotemporal graphs of human activities," in *ICCV*, 2011.
[12] 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.
[13] N. Quadrianto, A. J. Smola, L. Song, and T. Tuytelaars, "Kernelized sorting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 10, pp. 1809–1821, 2010.
[14] M. Zaslavskiy, F. R. Bach, and J.-P. Vert, "Global alignment of protein-protein interaction networks by graph matching methods," *Bioinformatics*, vol. 25, no. 12, pp. 1259–1267, 2009.
[15] 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.*, vol. 176, no. 2, pp. 657–690, 2007.
[16] R. Burkard, M. DellAmico, and S. Martello, *Assignment Problems*. SIAM, 2009.
[17] F. Zhou and F. De la Torre, "Factorized graph matching," in *CVPR*, 2012.
[18] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *Int. J. Pattern Recognit. and Artificial Intelligence*, vol. 18, no. 3, pp. 265–298, 2004.
[19] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, vol. 23, pp. 31–42, 1976.
[20] M. Pelillo, "Replicator equations, maximal cliques, and graph isomorphism," *Neural Comput.*, vol. 11, no. 8, pp. 1933–1955, 1999.
[21] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1367–1372, 2004.
[22] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 5, pp. 695–703, 1988.
[23] G. Scott and H. Longuet-Higgins, "An algorithm for associating the features of two images," *Biological Sciences*, vol. 244, no. 1309, pp. 21–26, 1991.
[24] L. S. Shapiro and M. Brady, "Feature-based correspondence: an eigenvector approach," *Image Vision Comput.*, vol. 10, no. 5, pp. 283–288, 1992.
[25] T. Caelli and S. Kosinov, "An eigenspace projection clustering method for inexact graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 4, pp. 515–519, 2004.
[26] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," *Econometrica*, vol. 25, no. 1, pp. 53–76, 1957.
[27] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *ICCV*, 2005.
[28] T. Cour, P. Srinivasan, and J. Shi, "Balanced graph matching," in *NIPS*, 2006.
[29] P. H. S. Torr, "Solving Markov random fields using semidefinite programming," in *AISTATS*, 2003.
[30] C. Schellewald and C. Schnörr, "Probabilistic subgraph matching based on convex relaxation," in *EMMCVPR*, 2005.
[31] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, no. 6, pp. 1115–1145, 1995.
[32] H. A. Almohamad and S. O. Duffuaa, "A linear programming approach for the weighted graph matching problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 5, pp. 522–525, 1993.
[33] M. Zaslavskiy, F. R. Bach, and J.-P. Vert, "A path following algorithm for the graph matching problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2227–2242, 2009.
[34] Z. Liu, H. Qiao, and L. Xu, "An extended path following algorithm for graph-matching problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1451–1456, 2012.
[35] E. L. Lawler, "The quadratic assignment problem," *Management Science*, vol. 9, no. 4, pp. 586–599, 1963.
[36] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 4, pp. 377–388, 1996.
[37] Y. Tian, J. Yan, H. Zhang, Y. Zhang, X. Yang, and H. Zha, "On the convergence of graph matching: Graduated assignment revisited," in *ECCV*, 2012.
[38] B. J. van Wyk and M. A. van Wyk, "A POCS-based graph matching algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1526–1530, 2004.
[39] M. Leordeanu, M. Hebert, and R. Sukthankar, "An integer projected fixed point method for graph matching and MAP inference," in *NIPS*, 2009.

[40] L. Torresani, V. Kolmogorov, and C. Rother, "Feature correspondence via graph matching: Models and global optimization," in *ECCV*, 2008.
[41] A. Egozi, Y. Keller, and H. Guterman, "A probabilistic approach to spectral graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 18–27, 2013.
[42] R. Zass and A. Shashua, "Probabilistic graph and hypergraph matching," in *CVPR*, 2008.
[43] M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching," in *ECCV*, 2010.
[44] M. Leordeanu, R. Sukthankar, and M. Hebert, "Unsupervised learning for graph matching," *Int. J. Comput. Vis.*, vol. 95, no. 1, pp. 1–18, 2011.
[45] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola, "Learning graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1048–1058, 2009.
[46] 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.*, vol. 33, no. 12, pp. 2383–2395, 2011.
[47] M. Cho and K. M. Lee, "Progressive graph matching: Making a move of graphs via probabilistic voting," in *CVPR*, 2012.
[48] A. Myronenko and X. B. Song, "Point set registration: Coherent point drift," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, 2010.
[49] O. van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or, "A survey on shape correspondence," *Comput. Graph. Forum.*, vol. 30, no. 6, pp. 1681–1707, 2011.
[50] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *3DIM*, 2001.
[51] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, *Numerical geometry of non-rigid shapes*. Springer, 2008.
[52] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, 2002.
[53] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Comput. Vis. Image Underst.*, vol. 89, no. 2-3, pp. 114–141, 2003.
[54] B. Luo and E. R. Hancock, "A unified framework for alignment and correspondence," *Comput. Vis. Image Underst.*, vol. 92, no. 1, pp. 26–55, 2003.
[55] Y. Zheng and D. S. Doermann, "Robust point matching for nonrigid shapes by preserving local neighborhood structures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 643–649, 2006.
[56] H. Bunke, "Graph matching: Theoretical foundations, algorithms, and applications," in *VI*, 2000.
[57] 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.
[58] T. S. Caetano, T. Caelli, D. Schuurmans, and D. Augusto, "Graphical models and point pattern matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1646–1663, 2006.
[59] J. Maciel and J. Costeira, "A global solution to sparse correspondence problems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 2, pp. 187–199, 2003.
[60] E. L. Allgower and K. Georg, *Introduction to Numerical Continuation Methods*. SIAM, 2003.
[61] A. L. Yuille and A. Rangarajan, "The concave-convex procedure," *Neural Comput.*, vol. 15, no. 4, pp. 915–936, 2003.
[62] R. Horst and P. M. Pardalos, *Handbook of Global Optimization*. Kluwer, 1995.
[63] J. Von Neumann, "A certain zero-sum two-person game equivalent to the optimal assignment problem," *Contributions to the Theory of Games*, vol. 2, pp. 5–12, 1953.
[64] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.
[65] M. Fukushima, "A modified Frank-Wolfe algorithm for solving the traffic assignment problem," *Transp. Res. Part B: Methodological*, vol. 18, no. 2, pp. 169–177, 1984.
[66] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. F. Tappen, and C. Rother, "A comparative study of energy minimization methods for Markov random fields with smoothness-based priors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 6, pp. 1068–1080, 2008.

Feng Zhou received the BS degree in computer science from Zhejiang University in 2005, the MS degree in computer science from Shanghai Jiao Tong University in 2008, and the PhD degree in robotics from Carnegie Mellon University in 2014. He is now a researcher in the Media Analytics group at NEC Laboratories America. His research interests include machine learning and computer vision.

Fernando de la Torre is an Associate Research Professor in the Robotics Institute at Carnegie Mellon University. He received his B.Sc. degree in Telecommunications, as well as his M.Sc. and Ph. D degrees in Electronic Engineering from La Salle School of Engineering at Ramon Llull University, Barcelona, Spain in 1994, 1996, and 2002, respectively. His research interests are in the fields of Computer Vision and Machine Learning. Currently, he is directing the Component Analysis Laboratory (http://ca.cs.cmu.edu) and the Human Sensing Laboratory (http://humansensing.cs.cmu.edu) at Carnegie Mellon University. He has over 150 publications in referred journals and conferences. He has organized and co-organized several workshops and has given tutorials at international conferences on the use and extensions of Component Analysis.