Deformable Graph Matching

Feng Zhou Fernando De la Torre
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213
http://www.f-zhou.com ftorre@cs.cmu.edu

Abstract

Graph matching (GM) is a fundamental problem in computer science, and it has been successfully applied to many problems in computer vision. Although widely used, existing GM algorithms cannot incorporate global consistence among nodes, which is a natural constraint in computer vision problems. This paper proposes deformable graph matching (DGM), an extension of GM for matching graphs subject to global rigid and non-rigid geometric constraints. The key idea of this work is a new factorization of the pair-wise affinity matrix. This factorization decouples the affinity matrix into the local structure of each graph and the pair-wise affinity edges. Besides the ability to incorporate global geometric transformations, this factorization offers three more benefits. First, there is no need to compute the costly (in space and time) pair-wise affinity matrix. Second, it provides a unified view of many GM methods and extends the standard iterative closest point algorithm. Third, it allows to use the path-following optimization algorithm that leads to improved optimization strategies and matching performance. Experimental results on synthetic and real databases illustrate how DGM outperforms state-of-the-art algorithms for GM. The code is available at http://humansensing.cs.cmu.edu/fgm.

1. Introduction

Graph matching (GM) has been widely applied in computer vision to solve a variety of problems such as object categorization [10], feature tracking [13, 17], symmetry analysis [12], kernelized sorting [20] and action recognition [3]. From an optimization view-point, the GM problem is typically formulated as a quadratic assignment problem (QAP) [18]. Unlike the linear assignment problem, which can be efficiently solved with the Hungarian algorithm [4], the QAP is known to be NP-hard and exact optimal algorithms using variations of branch-and-bound [22] are only practical for very small graphs (e.g., 30 nodes). Therefore, the main body of research in GM has focused on devising more accurate and faster algorithms to approximate it.

Although extensive research has been done on GM for decades, there are still two main challenges: (1) Many matching problems in computer vision naturally require global constraints among nodes in the graph. For instance, given two sets of coplanar points in two images, the matching between points should be constrained by an affine transformation (under orthographic projection). 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). (2) Optimizing GM is still difficult because the objective function is in general non-convex and the constraints are combinatorial. While there are a number of papers [6, 8, 11, 14, 24, 26, 15] addressing the second issue, the first has been rarely explored. This paper proposes DGM, an extension of GM that solves the first problem, and improves upon the second issue.

In order to incorporate global transformations, the key
idea of our method is to factorize the pairwise affinity matrix into matrices that preserve the local structure of each graph and matrices that encode the similarity between nodes and edges. This factorization is general and can be applied to both directed and undirected graphs. Consider the two graphs shown in Fig. 1 as an example. Using the factorization, we are able to factorize the large 20-by-20 pair-wise affinity matrix into six smaller matrices. Because we have decoupled the local structure for the nodes in each graph, it is easy to add global geometric constraints. Moreover, using this factorization has three additional benefits for GM. First, there is no need to compute the costly (in space and time) pair-wise affinity matrix. Second, it provides a unified view of many GM methods, which allows to understand the commonalities and differences between them. It also connects GM methods with the classical iterative closest point (ICP) algorithm, and provides a pair-wise generalization of ICP. Third, it allows the use of path-following optimization algorithms in general GM problems that leads to improved optimization strategies and matching performance. We illustrate the benefits of DGM in synthetic and real matching experiments on standard databases.

2. Previous works

2.1. Graph matching (GM)

We denote (see notation\(^1\)) a graph with \(n\) nodes and \(m\) directed edges as a 4-tuple \(G = \{P, Q, G, H\}\). The features for nodes and edges are specified by \(P = [p_1, \ldots, p_n] \in \mathbb{R}^{d_x \times n}\) and \(Q = [q_1, \ldots, q_m] \in \mathbb{R}^{d_q \times m}\) respectively. The topology of the graph is encoded by two node-edge incidence matrices \(G, H \in \{0, 1\}^{n \times m}\), where \(g_{ic} = h_{jc} = 1\) if the \(c\)th edge starts from the \(i\)th node and ends at the \(j\)th node. For instance, Fig. 2a illustrates two synthetic graphs, whose edge connection between nodes is encoded by the corresponding matrices shown in Fig. 2b–c. A similar representation of graph was adopted in [29]. However, the work in [29] is only valid for undirected graphs. Our representation is more general and valid for directed and undirected graphs. Directed graphs typically occur when the features are asymmetrical such as the angle between an edge and the horizontal line. Our model incorporates directed graphs by encoding the starting and ending node in \(G\) and \(H\) respectively.

Given two graphs, \(G_1 = \{P_1, Q_1, G_1, H_1\}\) and \(G_2 = \{P_2, Q_2, G_2, H_2\}\), we compute two affinity matrices, \(K_p \in \mathbb{R}^{n_1 \times n_2}\) and \(K_q \in \mathbb{R}^{m_1 \times m_2}\), to measure the similarity of each node and edge pair respectively. More specifically, \(\kappa^p_{i_1j_2} = \phi_p(p^{i_1}_1, p^{j_2}_2)\) measures the similarity between the \(i_1\)th node of \(G_1\) and the \(j_2\)th node of \(G_2\), and \(\kappa^q_{c_1c_2} = \phi_q(q^{c_1}_1, q^{c_2}_2)\) measures the similarity between the \(c_1\)th edge of \(G_1\) and the \(c_2\)th edge of \(G_2\). For instance, Fig. 2d illustrates an example pair of \(K_p\) and \(K_q\) for the two synthetic graphs.

It is more convenient to encode the node and edge affinities in a global affinity matrix \(K \in \mathbb{R}^{n_1 \times n_2 \times m_1 \times m_2}\), whose element is computed as follows:

\[
\kappa_{i_1j_2c_1c_2} = \begin{cases} 
\kappa^p_{i_1j_2}, & \text{if } i_1 = j_1 \text{ and } i_2 = j_2, \\
\kappa^q_{c_1c_2}, & \text{if } i_1 \neq j_1 \text{ and } i_2 \neq j_2 \text{ and } g_{i_1c_1} h_{j_2c_2} = 1, \\
0, & \text{otherwise.}
\end{cases}
\]

Given two graphs and \(K\), the problem of GM consists in finding the optimal correspondence \(X\) between nodes, such that the following score is maximized,

\[
\max_X J_{gm}(X) = \text{vec}(X)^T K \text{vec}(X), \quad \text{s. t. } X \in \mathbb{R}^{m_1 \times m_2}.
\]

where \(X \in \mathbb{P}\) is usually constrained to be a one-to-one mapping, i.e., \(\Pi\) is the set of partial permutation matrices:

\[
\Pi = \{X|X \in \{0, 1\}^{n_1 \times n_2}, X_{i_1n_2} \leq 1, n_1, X^T 1_{n_1} = 1_{n_2}\}.
\]

The inequality in the above definition is used for the case when the graphs are of different sizes. Without loss of generality, we assume \(n_1 \geq n_2\) throughout the rest of the paper.

Advances in GM: GM can be formulated as a quadratic assignment problem [18] and optimizing Eq. 1 is known to be NP-hard. Therefore, major research in GM has focused on finding better optimization strategies. Broadly speaking, most relaxations of the permutation constraints fall into two categories: spectral and doubly-stochastic.

The first group of methods approximates the permutation matrix with an orthogonal one, i.e., \(X^T X = I\). Under the orthogonal constraint, optimizing \(J_{gm}(X)\) can be solved in closed-form as an eigen-value problem [23, 21]. However, these methods can only work for a restricted case, where \(K = K_1 \otimes K_2\) is composed by two weighted adjacency matrices, \(K_1 \in \mathbb{R}^{n_1 \times n_1}\) and \(K_2 \in \mathbb{R}^{n_2 \times n_2}\), defined on each graph respectively. In order to handle more complex problems in computer vision, Leordeanu and Hebert [14] proposed to optimize Eq. 1 by relaxing the constraints on \(X\) to be of unit length, i.e., \(\|\text{vec}(X)\|_2^2 = 1\). In this case, the optimal X can be simply computed as the leading eigenvector of \(K\). Cour et al. [8] incorporated additional affine constraints to solve a more general spectral problem.

The second group of methods relaxes \(X \in \mathcal{D}\) to be a doubly stochastic matrix, the convex hull of \(X \in \mathbb{P}\),

\[
\mathcal{D} = \{X \in \mathbb{R}^{n_1 \times n_2} | X_{1n_2} \leq 1_{n_1}, X^T 1_{n_1} = 1_{n_2}, X \geq 0\}.
\]
under this constraint, optimizing Eq. 1 can be treated as a non-convex quadratic programming problem and various strategies have been proposed to find a local optimum. For instance, Gold and Rangarajan [11] proposed the graduated assignment algorithm to iteratively solve a series of linear approximations of the cost function using Taylor expansions. Leordeanu et al. [15] proposed an integer projection algorithm to optimize the objective function in an integer domain. More recently, Zhou and De la Torre [29] used a path-following algorithm [25]. In addition to the optimization-based work, probabilistic frameworks [6, 26] were shown to be useful for interpreting and solving GM.

Our work is closely related to recent higher-order tensor factorization [5, 9, 26]. It has been noticed that K encoding the pairwise geometry is susceptible to scale and rotation differences between sets of points. In order to make GM invariant to rigid deformations, [5, 9, 26] extended the pairwise matrix K embedded into 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 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. On the other hand, it is unclear on how to extend high-order methods to incorporate non-rigid deformations.

2.2. Iterative closest point (ICP)

Given two sets of points, \( P_1 = [p_1^1, \ldots, p_{n_1}^1] \in \mathbb{R}^{d \times n_1} \) and \( P_2 = [p_1^2, \ldots, p_{n_2}^2] \in \mathbb{R}^{d \times n_2} \), iterative closest point (ICP) algorithms (e.g., \([2, 27]\)) aim to find the correspondence and the geometric transformation between points such that the sum of distances is minimized:

\[
\min_{X, \mathcal{T}} J_{icp}(X, \mathcal{T}) = \sum_{i=1}^{n_1} \|p_i^1 - \mathcal{T}(p_i^2)\|_2^2 + \psi(\mathcal{T}),
\]

subject to \( X \in \Pi, \mathcal{T} \in \Psi \),

where \( X \in \{0, 1\}^{n_1 \times n_2} \) denotes the correspondence between points. Depending on the problem, \( X \) denotes either a one-to-one or many-to-one mapping. In this paper, we consider a one-to-one mapping between points and \( X \) is thus constrained to be a permutation matrix \( i.e., X \in \Pi \).

\( \tau(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d \) denotes a geometric transformation and it is parameterized by \( \mathcal{T} \). For instance, if \( \tau(\cdot) \) is a 2-D similarity transformation, then \( \tau(p) = sRp + t \) and \( \mathcal{T} = \{s, R, t\} \), where \( s \in \mathbb{R} \) is the scaling factor, \( R \in \mathbb{R}^{2 \times 2} \) is the rotation matrix and \( t \in \mathbb{R}^2 \) is the translation vector. In addition, the rotation matrix has to satisfy the constraint, \( \Psi = \{R \mid R^T R = I_{2}, |R| = 1\} \). If \( \tau(\cdot) \) is chosen to be a non-rigid transformation, a penalization cost \( \psi(\mathcal{T}) \) is needed to further constrain the parameter. See [28] for a more comprehensive review of various transformations adopted in ICP.

To connect ICP with GM methods, we re-write Eq. 2 as:

\[
J_{icp}(X, \mathcal{T}) = -\text{tr} \left(K_p(\mathcal{T})^T X\right) + \psi(\mathcal{T}).
\]

where \( K_p(\mathcal{T}) \in \mathbb{R}^{n_1 \times n_2} \) encodes the Euclidean distances between nodes, that is, \( K_{p_{i_1}^1, p_{i_2}^2}(\mathcal{T}) = -\|p_i^1 - \mathcal{T}(p_i^2)\|_2^2 \). Eq. 3 reveals two commonalities of ICP algorithms: (1) The optimization over X given the transformation \( \mathcal{T} \) can be cast as a linear matching problem, which can be efficiently optimized by the Hungarian algorithm (if \( X \) is a one-to-one mapping) or the winner-take-all manner (if \( X \) is a many-to-one mapping). (2) In general, the joint optimization over \( X \) and \( \mathcal{T} \) is non-convex, and no closed-form solution is known. Typically, some sort of alternating minimization (e.g., EM, coordinate-descent) is needed to find a local optima.

3. Factorized graph matching

This section derives a new factorization of the pair-wise affinity matrix K. As we will see in the following sections, this factorization allows the unification of GM methods, adding geometric constraints to GM and elaborating better optimization strategies.
To illustrate the intuition behind the factorization, let us consider the synthetic graph shown in Fig. 2. Notice that $K \in \mathbb{R}^{n_1 \times n_2}$ is composed by two types of affinities: the node affinity ($K_{ij}$) on its diagonal and the pairwise edge affinity ($K_{ij}$) on its off-diagonals. Let’s ignore the diagonal first. Then, $K$ is a sparse block matrix with three unique structures: (1) $K$ is composed by $n_2$-by-$n_2$ smaller blocks $K_{ij} \in \mathbb{R}^{n_1 \times n_1}$. (2) Some of the $K_{ij}$’s are empty if there is no edge connecting the $i^{\text{th}}$ and $j^{\text{th}}$ nodes of $G_2$. In another word, these empty blocks can be indexed by $G_2 H_2^T$, i.e., $K_{ij} = 0_{n_1 \times n_1}$ if $[G_2 H_2^T]_{ij} = 0$. (3) For the non-empty blocks, $K_{ij}$ can be computed in a closed form as $G_1 \text{diag}(k^2_i) H_1^T$, where $c$ is the index of the edge connecting the $i^{\text{th}}$ and $j^{\text{th}}$ nodes of $G_2$, i.e., $g^T_{ij} = h^T_{ij} = 1$. Based on these three observations, and after some linear algebra, it can be shown that $K$ can be exactly factorized as:

$$K = \text{diag} (\text{vec}(K_p)) + (G_2 \otimes G_1) \text{diag} (\text{vec}(K_q)) (H_2 \otimes H_1)^T.$$  

(4)

This factorization decouples the graph structure ($G_1$, $H_1$, $G_2$ and $H_2$) from the similarity ($K_p$ and $K_q$). It is important to notice that our factorization significantly differs from the one proposed in [29] in two aspects: (1) $K_p$ is proposed for more general graphs composed by directed edges while [29] can be only applied for simpler graphs composed by undirected edges; (2) Unlike a joint factorization proposed in [29], Eq. 4 separates $K_p$ and $K_q$ in the factorization in two independent terms. This separation enables us to introduce geometric transformations on $K_p$ and $K_q$ in GM.

Eq. 4 is the key contribution of this work. Previous work in GM computed the computationally expensive (in space and time) $K$. On the contrary, Eq. 4 offers an alternative framework by replacing $K$ with six smaller matrices. For instance, plugging Eq. 4 into Eq. 1 leads to an equivalent objective function:

$$J_{gm}(X) = \text{tr} (K_p^T X) + \text{tr} (K_q^T Y).$$  

(5)

where $Y = (G_1 X G_2 \circ H_1^T H_2 H_2^T X H_2) \in \{0, 1\}^{m_1 \times m_2}$ is an auxiliary variable that encodes the correspondence between edges, i.e., $y_{ij} = 1$ if $e_{ij}$ in $G_1$ is matched to the $e_{ij}$ edge in $G_2$. For instance, Fig. 2e illustrates the node and edge correspondence matrices for the matching defined in Fig. 2a. In addition, Eq. 5 reveals a connection between GM and ICP. In particular, maximizing the first term of Eq. 5 is equivalent to ICP (Eq. 3).

Observe that $K_q$ can always be factorized (e.g., SVD) as $K_q = U V^T$, where $U \in \mathbb{R}^{m_2 \times c}$ and $V \in \mathbb{R}^{m_2 \times c}$. Taking advantage of the low-rank structure of $K_q$, Eq. 5 can be further re-formulated as follows:

$$J_{gm}(X) = \text{tr} (K_p^T X) + \sum_{i=1}^c \text{tr} (A_i^1 X A_i^2 X^T),$$  

(6)

where $A_i^1 = G_1 \text{diag}(u_i) H_1^T$ and $A_i^2 = G_2 \text{diag}(v_i) H_2^T$.

The factorization (Eq. 4) and the two equivalent objectives (Eq. 5 and Eq. 6) allow to unify GM methods. For instance, Eq. 6 reveals the connection between two types of GM problems, the less general one [1, 23, 25] that maximizes $\text{tr}(A_1 X A_2 X^T)$, versus the more general one [6, 8, 11, 14, 15, 24, 26] that maximizes $\text{vec}(X)^T K \text{vec}(X)$. In particular, maximization of $\text{vec}(X)^T K \text{vec}(X)$ is equivalently to the maximization of the sum of $c$ traces $\text{tr}(A_i^1 X A_i^2 X^T)$, where $A_i^1$ and $A_i^2$ can be interpreted as adjacency matrices.

### 3.1. A path-following algorithm

Given Eq. 6 we can optimize GM with the path-following algorithm proposed for the simplified GM problem ($\text{tr}(A_1 X A_2 X^T)$) [1, 23, 25]. More specifically, we solved a series of concave-convex problems:

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

(7)

where $\alpha \in [0, 1]$ is a trade-off between the convex relaxation $J_{vex}(X)$ and the concave relaxation $J_{cav}(X)$ of the original objective $J_{gm}(X)$.

To employ the path-following algorithm, we need to find proper convex and concave relaxations of $J_{gm}(X)$. Fortunately, the factorization (Eq. 4) offers a principled way for deriving them:

$$J_{vex}(X) = J_{gm}(X) - \frac{1}{2} J_{con}(X) = \text{tr} (K_p^T X) - \frac{1}{2} \sum_{i=1}^c \|X^T A_i^1 - A_i^2 X^T\|_F^2,$$  

(8)

$$J_{cav}(X) = J_{gm}(X) + \frac{1}{2} J_{con}(X) = \text{tr} (K_p^T X) + \frac{1}{2} \sum_{i=1}^c \|X^T A_i^1 + A_i^2 X^T\|_F^2,$$  

(9)

$$J_{con}(X) = \sum_i \text{tr} (A_i^1 X H_2 X A_i^2)^T + \text{tr} (A_i^2 X H_2 A_i^1)^T,$$

where $J_{con}(X) = \gamma$ is a constant with respect to a permutation or orthogonal matrix $X$ because $XX^T = X^T X = I$. It is worth to point out that it not clear how to derive the relaxations ($J_{vex}(X)$ and $J_{cav}(X)$) and apply the path-following algorithm without the propose factorization of $K$. Please refer [28] for details about the path-following optimization.

The advantages of the path-following algorithm over conventional GM algorithms are three-fold: (1) The algorithm starts with a convex problem ($\alpha = 0$) and it is guaranteed to find a globally optimal solution. (2) The algorithm ends at a concave problem ($\alpha = 1$) and the local optimal solution is always discrete; (3) By smoothly increasing $\alpha$ from $0$ to $\alpha = 1$, the path-following algorithm is more likely to find better local optima than gradient-based method.
4. Deformable graph matching (DGM)

This section describes how to incorporate rigid and non-rigid transformation into the GM framework. Moreover, we illustrate how the factorization can be used into the DGM to derive an improved optimization strategy.

4.1. Objective function

To simplify the discussion and to be consistent with ICP, we compute the node feature of each graph \( G = \{ P, Q, G, H \} \) simply as the node coordinates, \( P = [p_1, \ldots, p_n] \in \mathbb{R}^{d \times n} \). Similarly, the edge features \( Q = [q_1, \ldots, q_m] \in \mathbb{R}^{d \times m} \) are computed as the coordinate difference between the connected nodes, i.e., \( q_{ic} = p_i - p_j \), where \( g_{ic} = h_{jc} = 1 \). In this case, the edge feature can be conveniently computed in a matrix form as, \( Q = P(G - H) \).

Suppose that we are given two graphs, \( G_1 = \{ P_1, Q_1, G_1, H_1 \} \) and \( G_2 = \{ P_2, Q_2, G_2, H_2 \} \), and a geometrical transformation defined on points by \( \tau(\cdot) \). Similar to ICP, we compute the node affinity \( K_{\tau}(T) \in \mathbb{R}^{n_1 \times n_2} \) and the edge affinity \( K_{\psi}(T) \in \mathbb{R}^{m_1 \times m_2} \) as a function of the Euclidean distance, i.e.:

\[
\kappa_{1,2}^n(T) = -\frac{1}{2} \| p_1 - \tau(p_2) \|^2, \\
\kappa_{1,2}^l(T) = \beta - \frac{1}{2} \| (p_1 - q_1^l) - \frac{\tau(q_2^l)}{\tau(q_2^l)} \|^2, \\
\tag{10}
\]

where \( \beta \) is chosen to be reasonably large to ensure that the pairwise affinity is greater than zero.

Recall that the factorization (Eq. 4) reveals that the goal of GM (Eq. 5) is similar to ICP (Eq. 3). In order to make the GM more robust to geometric deformations, DGM aims to find the optimal correspondence \( X \) as well as the optimal transformation \( T \) such that the global consistency can be maximized:

\[
\max_{X,T} J_{dgm}(X, T) = \text{tr} \left( K_{\tau}(T)^T X \right) + \lambda \text{tr} \left( K_{\psi}(T)^T Y \right) - \psi(T), \\
\text{s.t. } X \in \Pi, T \in \Psi,
\tag{11}
\]

where \( \lambda \geq 0 \) is used to balance between the importance of the node and edge consistency. Similar to ICP, \( \psi(T) \) and \( \Psi \) are used to constrain the transformation parameter. Eq. 11 unifies GM and ICP. In particular, if \( \lambda = 0 \), solving DGM is equivalent to ICP. In other case when \( \lambda > 0 \) and \( T \) is known, solving DGM is identical to a GM problem.

Due to the non-convex nature of the objective, we optimize DGM by alternatively solving the correspondence \( (X) \) and the transformation parameter \( (T) \). The initialization is important for the performance of DGM. However, the way of choosing a good initialization is beyond the scope of this paper and we simply set the initial transformation as an identity one, i.e., \( \tau(p) = p \).

4.2. Optimization

Optimizing Eq. 11 will alternate between optimizing for the correspondence and the geometric transformation.

**Optimization for the correspondence**: Given the transformation \( T \), DGM is equivalent to a traditional GM problem. To find the node correspondence \( X \), we adopt the path-following algorithm by optimizing Eq. 7.

**Optimization for the geometric transformation**: Given the correspondence matrix \( X \), the optimization over the transformation parameter \( T \) is similar to ICP. The main difficulty lies in the fact that the transformation parameter \( T \) appears not only in the node affinity \( K_{\psi}(T) \), but also in the edge affinity \( K_{\psi}(T) \). After some linear algebra, however, it can be shown that for certain choices of transformations in 2-D (e.g., similarity, affine, RBF non-rigid), the parameter can be computed in closed-form. For instance, let \( P_1 = P_1 - P_1^T X_1 \in \mathbb{R}^{2 \times n_1} \) and \( P_2 = P_2 - P_2^T X_1 \in \mathbb{R}^{2 \times n_2} \) be the centralized point sets, where \( p_1 = P_1 \), \( P_2 \) and \( P_2 \) are the mean vectors of the two point sets respectively. Then the parameters for the 2-D similarity transformation could be computed as:

\[
t = P_1 - s R P_2, \quad R = U \text{diag}(1, \cdots, |U V^T|) V^T, \\
\text{tr}(\Sigma) = \text{tr} \left( I_{n_1 \times 2}(P_2 \circ P_2) X^T \right) + \lambda \text{tr} \left( I_{n_1 \times 2}(Q_2 \circ Q_2) Y^T \right),
\]

where \( U \Sigma V^T = \tilde{P}_1 \tilde{X}^T \) is computed by SVD. Please refer [28] for the derivation of the optimal affine and non-rigid transformations.

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 pair-wise constrains, DGM is less sensitive to the initialization. Fig. 3a illustrates the problem of aligning two fish shapes under varying values for the initial rotation and scale parameters. As shown in Fig. 3b, ICP gets trapped into a local optima if the orientation gap is larger than \( \frac{1}{2} \pi \) (the error should be 0). Similarly, DGM fails for large orientation gap after two iterations (the left column of Fig. 3c). However, as the number of iterations increases, 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. 3c). This experiment shows that adding pairwise constraints can make the ICP algorithm more robust to the problem of local optima.

5. Experiments

This section reports experimental results on three benchmark datasets and compares FGM for Directed graphs (FGM-D) and DGM to several state-of-the-art methods for GM and ICP respectively. The first two experiments compare the path-following algorithm to other GM approaches.
is the pairwise distance between the connected nodes. We repeated \( \sim \) ground-truth correspondences. We computed for each node the feature, \( p_i \), which is the orientation of the normal vector to the contour. We adopted the Delaunay triangulation to build the graph. In this experiment, we consider the most general graph where the edge is directed and the edge feature is asymmetrical. More specifically, each edge was represented by a couple of values, \( q_c = [d_c, \theta_c]^T \), where \( d_c \) is the pairwise distance between the connected nodes and \( \theta_c \) is the angle between the edge and the horizontal line. Thus, for each pair of images, we computed the node affinity as \( k_{ij}^p = \exp(-\sum_{p_i} p_i - p_j) \) and the edge affinity as \( k_{c1c2}^q = \exp(-\frac{1}{2}[d_{c1} - d_{c2}]^T - \frac{1}{2}([\theta_{c1} - \theta_{c2}]^T) \). Fig. 5a and Fig. 5b 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 eight state-of-the-art methods. However, we were unable to directly use FGM-U to match directed graphs. Therefore, we ran FGM-U on an approximated undirected graph, where for each pair of directed edges, we computed its new edge affinity as the average value of the original ones.

As observed in Fig. 5c-d, the proposed FGM-D consistently outperformed other methods in both datasets. As we show in the previous experiment, the path-following algorithm used by FGM-D provides a better optimization strategy than existing approaches. On the other hand, although FGM-U has a similar path-following strategy, it did not perform well because it is only applicable to undirected edges. 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.

5.3. Fish and character shape dataset

The UCF shape dataset 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 under different deformations and outliers. In the first series of experiments, we rotated the template with a varying degree (between 0 and \( \pi \)). In the second set of experiments, a varying amount of outliers (between 0 and 20) were randomly added in the bounding box of template. For instance, Fig. 6a-b illustrate two pairs of example shapes with 20 outliers. We repeated...
Figure 4. 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.

Figure 5. Comparison of GM methods on 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.

Figure 6. 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 $\frac{3}{4}\pi$ 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.

As shown in Fig. 6c-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 $\frac{3}{4}\pi$. When the number of outliers increases, DGM can still match most points under large rotation at $\frac{3}{4}\pi$. In contrast, ICP and CPD drastically failed in presence of outliers and large rotations (Fig. 6e-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 [7]. To generate the nonrigid transformation, we followed a similar setting in [19], 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. Fig. 7a illustrates a synthetic pair of graphs.

We compared DGM with other two state-of-the-art GM methods: SM [14] and RRWM [6]. 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. 7b-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. DGM significantly improved FGM-D by estimating the transformation.

5.4. Conclusions

This paper proposes DGM, an extension of GM for matching points under a global geometric transformation for directed and undirected graphs. The key idea for DGM is a novel factorization of the pairwise affinity matrix. Sev-
geral benefits follow from the factorization. First, it avoids the expensive (in space and time) computation of the pairwise affinity matrix. Second, it allows for a unification of GM methods and provides a clean connection with existing ICP algorithms. Finally, the decomposition enables the use of path-following algorithms that improve the performance of GM methods.

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

References


